



燕山大学
YANSHAN UNIVERSITY

本科毕业设计（论文）

论文题目 基于知识图谱的足球知识问答系统

作者姓名 米亦宸

专 业 软件工程

指导教师 王晓妍副教授

2024 年 6 月

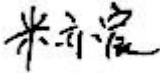
燕山大学本科毕业设计（论文）

基于知识图谱的足球知识问答系统

| | | |
|----------|------------|--------------|
| 学 | 院： | 信息科学与工程学院 |
| 专 | 业： | 软件工程 |
| 姓 | 名： | 米亦宸 |
| 学 | 号： | 202011200064 |
| 指 导 教 师： | 王晓妍 | |
| 答 辩 日 期： | 2024 年 6 月 | |

学位论文原创性声明

郑重声明：所提交的学位论文《基于知识图谱的足球知识问答系统》，是本人在导师的指导下，独立进行研究取得的成果。除文中已经注明引用的内容外，本论文不包括他人或集体已经发表或撰写过的作品成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果，并承诺因本声明而产生的法律结果由本人承担。

学位论文作者签名  日期：2024 年 5 月 30 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权燕山大学将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保 密 ☐，在__ 年解密后适用本授权书。

本学位论文属于

不保密 ☐。

(请在以上相应方框内打“√”)

学位论文作者签名：  日期：2024 年 5 月 30 日

指导教师签名：  日期：2024 年 5 月 30 日

附录 1 燕山大学毕业设计(论文)任务书

学院：信息科学与工程学院（软件学院）

系级教学单位： 软件工程系

| | | | | | |
|---|--|--|--|---------------------------|------------|
| 学号 | 20201120064 | 学生姓名 | 米亦宸 | 专业班级 | 20 级软件 3 班 |
| 题目 | 题目名称 | 基于知识图谱的足球知识问答系统 | | | |
| | 题目性质 | 1.理工类：工程设计（ ）；工程技术实验研究型（ ）；理论研究型（ ）；计算机软件型（ <input checked="" type="checkbox"/> ）；综合型（ ）。 2.文管类（ ）；3.外语类（ ）；4.艺术类（ ）。 | | | |
| | 题目类型 | 1.毕业设计（ <input checked="" type="checkbox"/> ） 2.论文（ ） | | | |
| | 题目来源 | 科研课题（ ） 生产实际（ ） 自选题目（ <input checked="" type="checkbox"/> ） | | | |
| 主要内容 | 这篇毕业论文旨在探讨和开发一个基于知识图谱的足球知识问答系统。首先，在文献综述部分，回顾相关领域的研究，包括自然语言处理、知识图谱和智能问答系统，同时考察现有足球知识图谱的应用和研究。接着，在方法与系统架构部分，将详细描述系统的整体架构，包括数据采集、知识图谱构建和自然语言处理组件，并解释如何整合足球相关数据，如球员、球队和比赛结果，以构建知识图谱。深入研究知识图谱的构建过程，包括数据抽取、清洗、实体关系建模和图谱存储，以及采用的图数据库和知识表示方法。 | | | | |
| 基本要求 | 理工类毕业设计要求学生必须基于专业知识，对本课题要解决的问题在社会环境下进行合理分析，评价对社会、健康、安全、文化以及可持续发展的影响，以及不同方案的优缺点和经济成本分析。（1）要求在开题前提出两套以上拟解决方案，经开题环节后确定最优方案；（2）要求学生必须基于专业知识，对本课题要解决的问题在社会环境下进行合理分析，评价对社会、安全、文化以及可持续发展的影响，以及不同方案的优劣和经济成本分析；（3）按照软件开发的基本流程，如定义、开发、实施、测试等环节实施该题目，并形成相应的过程文档；（4）标识符命名、代码格式应规范，并有必要的注释文字。（5）要求界面美观大方、操作简便，系统运行稳定可靠，易于维护和扩展。 | | | | |
| 参考资料 | [1] 基于时序知识图谱的足球知识问答系统. 方拓迁.广州大学,2021 | | | | |
| 周次 | 1—3 周 | 4—9 周 | 10—13 周 | 14—15 周 | |
| 应完成的内容 | 1、完成系统需求分析 2、广泛搜集和阅读相关资料，了解项目背景和研究意义 3、搭建开发环境;4、完成开题报告;5、2-3 周开题。 | 1、完成系统总体设计,进行系统详细设计,包括整体架构、接数据库设计、并编写相关文档。2、对系统进行编码实现。3、第 8-9 周中期检查 | 1、继续进行系统开发、测试与完善 2、结合设计文档及开发过程、中期检查结果等完善毕业论文，并提交。3、撰写论文初稿。 | 系统完善、论文撰写与完善、答辩用 PPT 制作等。 | |
| <p>指导教师： 王晓妍</p> <p>职称： 副教授</p> <p style="float: right;">2023 年 12 月 17 日</p> | | | | | |

摘 要

随着人工智能和自然语言处理技术的发展，知识图谱成为了一种强大的工具，能够在各种领域中提供智能问答服务。本文探讨了构建一个基于知识图谱的足球知识问答系统的方法。系统通过收集和整合足球相关的数据，构建出一个全面的足球知识图谱。然后，通过自然语言处理技术，实现对用户提出的足球问题的智能回答。实验结果表明，该系统在准确性和响应时间上都有良好的表现，证明了知识图谱在足球知识问答中的有效性。

系统架构设计包括数据收集与预处理模块、知识图谱构建模块、问答系统模块和用户界面模块。数据来源包括足球赛事数据库、球员和球队的官方网站、体育新闻和报道、专业足球统计网站等。数据经过清洗、格式转换和语义标注，确保数据的一致性和完整性。实体识别与关系抽取方面，系统识别关键足球实体如球员、球队、比赛、联赛等，并提取实体之间的关系。

利用 Neo4j 数据库存储构建好的知识图谱，定义节点类型和关系类型，并进行图谱存储。自然语言处理模块负责解析用户提出的问题，识别出问题中的关键实体和意图，并将用户问题转换为 Cypher 查询语句。系统利用预训练的语言模型（如 BERT、GPT）进行语义解析，提取出问题中的关键信息。

问答推理机制包括查询执行、答案生成和结果反馈。系统在 Neo4j 数据库中执行生成的 Cypher 查询语句，检索相关的知识图谱信息，并将查询结果转换为自然语言答案。实验与评估部分，设计了一系列实验验证系统的有效性和性能，结果表明系统在准确性和响应时间上表现良好，能够高效、准确地回答用户提出的足球问题。

本文研究表明，基于知识图谱的足球知识问答系统在数据组织和管理方面具有显著优势，通过结合大语言模型和知识图谱的技术，能够提升问答系统的准确性和响应速度，为用户提供更优质的服务。

关键词：足球；知识图谱；自然语言处理；系统设计；智能问答系统；

Abstract

With the development of Artificial Intelligence and Natural Language Processing technologies, Knowledge Graph has become a powerful tool to provide intelligent Q&A services in various domains. In this paper, we explore the method of building a soccer knowledge quiz system based on knowledge graph. The system constructs a comprehensive soccer knowledge graph by collecting and integrating soccer-related data. Then, it realizes intelligent answers to soccer questions asked by users through natural language processing technology. Experimental results show that the system performs well in terms of accuracy and response time, proving the effectiveness of knowledge graph in soccer knowledge quiz.

The system architecture design includes data collection and preprocessing module, knowledge graph construction module, Q&A system module and user interface module. Data sources include soccer event databases, official websites of players and teams, sports news and reports, and professional soccer statistics websites. The data are cleaned, format converted and semantically labeled to ensure the consistency and integrity of the data. For entity identification and relationship extraction, the system identifies key soccer entities such as players, teams, matches, leagues, etc., and extracts the relationships between the entities.

The Neo4j database is utilized to store the constructed knowledge graph, define node types and relationship types, and perform graph storage. The Natural Language Processing module is responsible for parsing the questions posed by the user, identifying the key entities and intents in the questions, and converting the user questions into Cypher query statements. The system utilizes pre-trained language models (e.g., BERT, GPT) for semantic parsing to extract the key information in the question.

The question and answer reasoning mechanism includes query execution, answer generation and result feedback. The system executes the generated Cypher query statements in the Neo4j database, retrieves relevant knowledge graph information, and converts the query results into natural language answers. In the Experiment and

Evaluation section, a series of experiments are designed to verify the effectiveness and performance of the system, and the results show that the system performs well in terms of accuracy and response time, and is able to efficiently and accurately answer soccer questions posed by users.

This paper shows that the soccer knowledge Q&A system based on knowledge graph has significant advantages in data organization and management, and by combining the techniques of large language model and knowledge graph, it can improve the accuracy and response time of the Q&A system, and provide users with better services.

Keywords: soccer; knowledge graph; natural language processing; system design; intelligent Q&A system;

目 录

| | |
|-----------------------------------|-----|
| 学位论文原创性声明 | I |
| 燕山大学毕业设计(论文)任务书 | III |
| 摘 要 | V |
| Abstract | VII |
| 第 1 章 绪 论 | 1 |
| 1.1 足球知识问答系统的背景与国内外研究动态 | 1 |
| 1.1.1 足球知识问答系统的背景与发展现状 | 1 |
| 1.1.2 技术背景与发展 | 1 |
| 1.1.3 国内外研究动态 | 2 |
| 1.2 本文研究内容 | 2 |
| 1.3 可行性分析 | 4 |
| 1.3.1 技术可行性 | 4 |
| 1.3.2 经济可行性 | 5 |
| 1.3.3 操作可行性 | 5 |
| 1.4 研究步骤、方法及措施 | 5 |
| 1.5 论文组织架构 | 7 |
| 1.6 本章小结 | 8 |
| 第 2 章 系统需求分析 | 9 |
| 2.1 研究内容 | 9 |
| 2.2 项目目标与范围定义 | 9 |
| 2.2.1 项目目标 | 9 |
| 2.2.2 范围定义 | 10 |
| 2.3 功能性需求 | 10 |
| 2.4 本章小结 | 12 |
| 第 3 章 基于知识图谱的足球知识问答系统的构建与推理 | 13 |
| 3.1 引言 | 13 |
| 3.2 系统架构设计 | 13 |
| 3.3 知识图谱的构建 | 13 |
| 3.3.1 数据收集与预处理 | 14 |
| 3.3.2 实体识别与关系抽取 | 15 |

| | |
|--------------------------------------|----|
| 3.3.3 图谱存储 | 16 |
| 3.4 自然语言处理 | 18 |
| 3.4.1 问题理解 | 18 |
| 3.4.2 查询生成 | 18 |
| 3.5 问答推理机制 | 19 |
| 3.5.1 查询执行 | 19 |
| 3.5.2 答案生成 | 21 |
| 3.5.3 结果反馈 | 21 |
| 3.5.3 实验与评估 | 22 |
| 3.7 本章小结 | 22 |
| 第 4 章 系统实现 | 23 |
| 4.1 基本服务介绍 | 23 |
| 4.2 详细设计原理与成果介绍 | 24 |
| 4.2.1 Neo4j 数据库设计 | 24 |
| 4.2.2 本地大语言模型 | 26 |
| 4.2.3 Bot 项目 | 28 |
| 4.2.4 使用 LLMs 为 neo4j 提供自然语言接口 | 29 |
| 4.2.5 前后端分离项目 | 31 |
| 4.6 本章小结 | 32 |
| 第 5 章 前后端分离项目 | 33 |
| 1.1 系统架构 | 33 |
| 5.1.1 前端架构 | 33 |
| 5.1.2 后端架构 | 33 |
| 5.1.3 获得回答接口 | 34 |
| 5.1.4 qa 接口 | 36 |
| 5.1.3 缓存机制 | 36 |
| 5.2 模块设计 | 36 |
| 5.2.1 首页展示界面 | 36 |
| 5.2.2 问答界面 | 37 |
| 5.2.3 RAG 模式支持 | 38 |
| 5.2.3 上传界面 | 40 |
| 5.4 本章小结 | 42 |
| 第 6 章 系统部署与应用 | 45 |
| 6.1 系统部署环境与工具 | 45 |
| 6.1.1 服务器相关信息 | 45 |

目 录

| | |
|------------------------|----|
| 6.1.2 服务端部署工具 | 45 |
| 6.1.3 客户端部署工具 | 46 |
| 6.2 构建与部署的实现过程 | 46 |
| 6.2.1 项目构建 | 47 |
| 6.2.2 Docker 的配置 | 47 |
| 结 论 | 51 |
| 参考文献 | 53 |
| 致 谢 | 55 |
| 附录 1 开题报告 | 57 |
| 附录 2 中期报告 | 69 |
| 附录 3 外文原文 | 83 |
| 附录 4 外文翻译 | 99 |

第 1 章 绪 论

1.1 足球知识问答系统的背景与国内外研究动态

1.1.1 足球知识问答系统的背景与发展现状

在现代社会，足球不仅是一项广受欢迎的体育运动，也是一个巨大的信息源。无论是球迷、媒体记者还是体育研究者，都需要及时、准确地获取足球相关的信息。传统的搜索引擎虽然能够提供大量的信息，但却往往缺乏针对性和准确性，用户需要花费大量时间来筛选和整理信息。知识图谱作为一种新兴的技术，能够有效地组织和管理大量结构化和非结构化数据，为智能问答系统的开发提供了新的可能。

足球是全球最受欢迎的体育运动之一，拥有数十亿的粉丝和庞大的市场。根据国际足联（FIFA）的统计数据，2018 年世界杯吸引了超过 35 亿观众，几乎占全球人口的一半。如此庞大的受众群体对足球信息的需求也是巨大的，不仅包括比赛结果、球员转会、战术分析，还包括历史数据和深度分析。

1.1.2 技术背景与发展

随着人工智能技术的迅速发展，知识图谱和自然语言处理领域特别是问答系统得到了显著的进步。知识图谱的本质是连接实体间关系的图，即揭示实体之间关系的语义网络，知识图谱作为一种组织和管理知识的有效方法，能够将世界上的事物及其相互关系以图形的方式表现出来，为问答系统提供了丰富的、结构化的知识来源^[8]。同时，近年来大语言模型，尤其是基于 Transformer architecture 的模型，如 GPT 系列和 BERT 系列，已经在自然语言理解和生成方面取得了突破性的成就。

大语言模型虽然能够生成流畅的语言回答，但在准确性和可靠性方面往往依赖于训练数据的质量和范围，LLM 通过概率模型进行推理，这是一个犹豫不决的过程。LLM 用于预测或决策的具体模式和函数对人类来说是不可直接访问或解释的。即使一些 LLM 能够通过应用思维链来解释其预测结果，它的推理解释会出现错觉问题^[3]，这就是知识图谱发挥作用的地方。足球知识问答系统作为一个具体应用领域，利用

知识图谱处理和回答与足球相关的各种问题，包括球员信息、比赛结果、历史记录等。

1.1.3 国内外研究动态

随着国家在体育事业上不断地投入以及体育锻炼强身健体这一观念的深入人心，人参与体育运动的积极性也越来越高，其中尤其以足球这项运动受众最广，参与程度最高。足球运动因其观赏性，对抗性以及不可预知性，成为了世界第一运动。

然而上述常见的知识图谱项目不是专门面向足球领域，对足球领域知识只有少量涉及。因为足球数据涉及大量的时间相关信息，如比赛日期、球员转会时间等。如何在知识图谱中表示这些时间信息，就成了一个挑战。

A Saxena 等人探讨了如何使用时间知识图谱来回答自然语言问题，通过在知识图谱的每条边上提供时间范围来拓展常规知识图^[7]，方拓迁提出时序知识图谱的表示模型 STKM，用于解决常规知识图谱表示模型无法表示时序知识的问题，但为了给知识图谱中的三元组添加时间信息，使得时序知识图谱产生了大量的无实际意义但不可缺少的三元组造成了时序知识图谱存储的有效信息的密度下降。

大语言模型虽然能够生成流畅的语言回答，但在准确性和可靠性方面往往依赖于训练数据的质量和范围，LLM 通过概率模型进行推理，这是一个犹豫不决的过程。LLM 用于预测或决策的具体模式和函数对人类来说是不可直接访问或解释的。即使一些 LLM 能够通过应用思维链来解释其预测结果，它的推理解释会出现错觉问题^[3]，知识图谱提供了结构化、明确和可编辑的知识表征，为缓解 LLM 的局限性提供了一种补充策略。研究人员已经探索了如何使用 KG 作为外部知识源来减轻 LLM 的幻觉。

Li 等使用 LLM 生成 SPARQL 查询的主干，并使用知识图谱填充完整的信息^[5]，Back 等人采样了包含出现问题的实体的三元组，用于 LLM 推理^[4]，Wang 等人提出了一种检索器-阅读器-验证器 QA 系统，用于访问外部知识并与 LLM 交互。

1.2 本文研究内容

本研究将实现一个基于基于知识图谱的足球知识问答系统，利用知识图谱与大语言模型相结合，利用两者技术优势建立出解决知识图谱时间处理困难的问题和大语言模型出现幻读的问题^[3]，同时与 Web 技术相结合，实现处理和回答与足球相关

的各种问题，包括球员信息、比赛结果、历史记录等。

该研究拟解决的主要问题包括：

(1) 知识图谱时间处理困难：

足球知识问答系统需要处理大量与时间相关的数据，如比赛日期、球员转会时间等。知识图谱在处理这类时间敏感信息时面临着挑战，如时间信息的表示、存储和查询等。研究将探索有效的时间信息处理方法，以提高知识图谱在足球领域的应用效率和准确性。

(2) 查询方式难：

传统的足球问答系统可能要求用户了解特定的查询语法或者使用精确的关键词来提出问题，这对于非技术用户来说可能既困难又不直观。足球数据非常丰富和多样，涵盖比赛结果、球员统计、转会信息等多个维度，传统查询方式难以有效处理这些复杂的数据关系和多维度的信息需求。

(3) 大语言模型的幻读问题：

在利用大语言模型进行问答时，可能会遇到基于错误或过时信息生成答案的问题，这种现象被称为幻读^[10,11]。研究将探讨如何减少幻读现象，通过结合知识图谱提供的结构化、可靠信息，提高问答系统的准确性和可靠性，如图 1-1 未使用来自知识图谱的信息，前者表示无法获取正确答案，而后者通过一个预处理的信息，可以得到一个准确的回复。

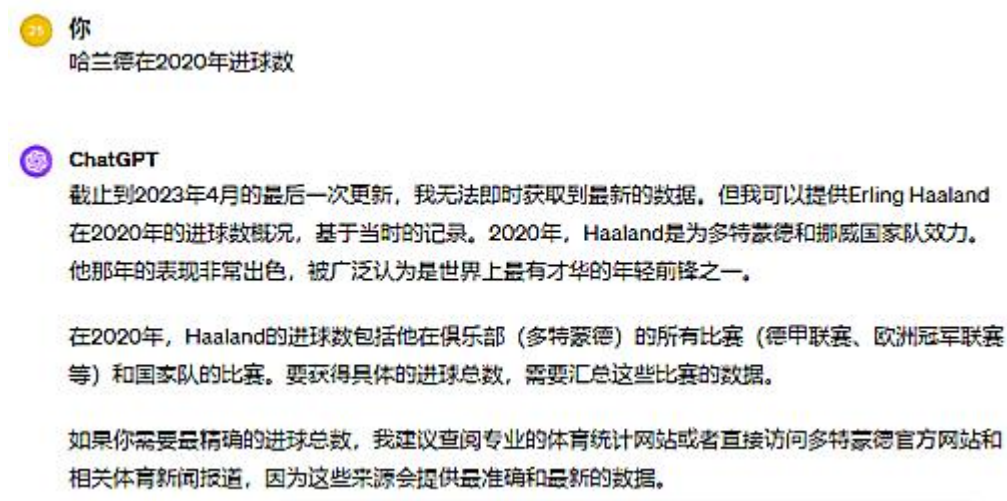


图 1-1 未使用来自知识图谱的信息

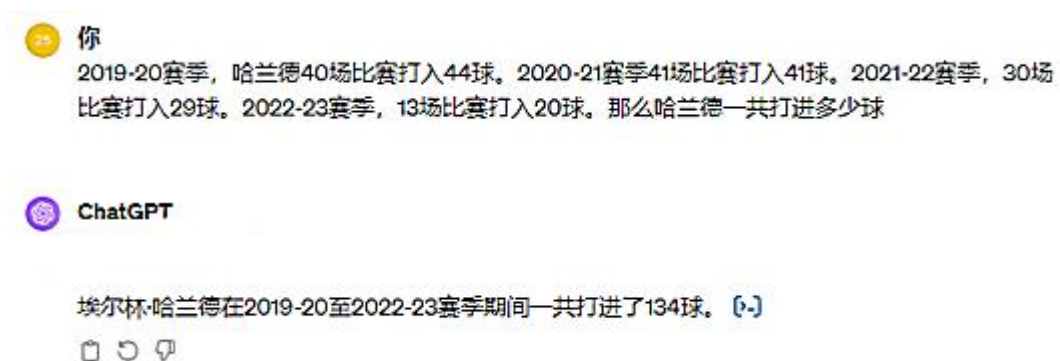


图 1-2 使用来自知识图谱的信息

(1) 知识图谱与大语言模型的有效结合：

如何有效地结合知识图谱和大语言模型的技术优势，是本研究的核心问题之一。研究将探索两者之间的交互机制，如利用知识图谱增强大语言模型的语义理解能力，同时利用大语言模型弥补知识图谱在自然语言处理方面的不足。

(2) 与 Web 技术的整合：

为了实现足球知识问答系统的广泛可访问性和用户友好性，研究还将探讨如何将知识图谱和大语言模型与 Web 技术相结合，包括前端界面设计、后端数据处理、以及用户交互方式等，以提供一个易于使用、响应迅速的在线问答平台。

1.3 可行性分析

1.3.1 技术可行性

基于知识图谱的足球知识问答系统在技术上具有很高的可行性，主要表现在以下几个方面：

成熟的技术框架：系统采用了成熟的技术框架，如 Python、Neo4j 数据库、LangChain 和 Streamlit 等。这些技术框架在各自的领域都有广泛的应用和社区支持，能够保证系统的稳定性和性能。

现有的自然语言处理技术：使用预训练的语言模型（如 GPT-4、BERT）能够高效地解析和理解自然语言问题，这些模型在多个实际应用中已经证明了其有效性和准确性。

强大的图数据库：Neo4j 作为一种流行的图数据库，具备高效的图数据存储和查

询能力，能够很好地支持复杂关系数据的管理和检索，适用于知识图谱的构建和查询^[12,13]。

1.3.2 经济可行性

经济可行性分析主要考虑系统开发、运行和维护的成本效益：

开发成本：系统所需的技术和工具大多为开源软件（如 Python、Neo4j），可以大幅降低软件许可成本。开发团队只需投入人力成本来实现系统的具体功能。

运行成本：系统可以部署在常见的云计算平台（如 AWS、Azure、Google Cloud）上，这些平台提供了灵活的计费方式，可以根据实际使用情况调整资源配置，控制运行成本。

维护成本：系统采用模块化设计，各组件独立运行，维护起来相对简单。同时，利用开源社区的资源和支持，可以降低维护难度和成本。

1.3.3 操作可行性

操作可行性分析主要评估系统在实际应用中的可操作性和用户体验：

用户界面友好：使用 Streamlit 框架构建的用户界面简洁直观，用户可以方便地提出问题并查看答案，无需复杂的操作步骤。

高效的数据处理：系统具备自动化的数据收集和处理能力，能够高效地构建和更新知识图谱，确保数据的实时性和准确性。

灵活的扩展性：系统设计具有良好的扩展性，可以方便地集成新的数据源和功能模块，满足未来不断变化的需求。此外，系统采用 Docker 和 Docker Compose 进行部署和管理，简化了环境配置和服务管理，提升了操作便捷性。

1.4 研究步骤、方法及措施

(1) 知识图谱构建

首先需要构建或使用现有的涵盖足球相关信息的知识图谱。这可能包括实体如球员、球队、联赛、比赛、比分等。同时也需要定义这些实体之间的关系，例如“效力于”、“是…的成员”、“参加了”等，如图 1-3 所示。

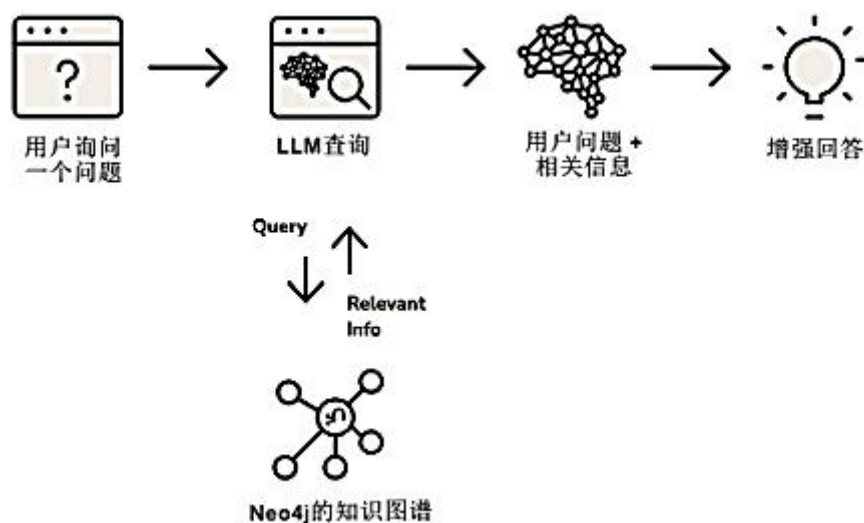


图 1-3 使用 LLMs 应用查询知识图谱

(2) 数据来源

从多种数据源中填充知识图谱，包括体育数据库、足球协会的官方网站、新闻报道、比赛统计和历史记录等。

(3) 数据处理

通过知识抽取技术，可从已有的结构化、半结构化、非结构化样本源以及一些开源的百科类网站抽取实体、关系、属性等知识要素。通过知识融合，可消除实体、关系、属性等指称项与实体对象之间的歧义，得到一系列基本的事实表达。通过本体抽取、知识推理和质量评估形成最终的知识图谱库。按照知识抽取、知识融合、知识推理 3 个步骤对知识图谱迭代更新，实现碎片化的互联网知识的自动抽取、自动关联和融合、自动加工，从而拥有词条自动化链接、词条编辑辅助功能，最终达成全流程自动化知识获取的目标。

(1) 图谱查询

根据用户的问题，系统需要能够构建和执行针对知识图谱的查询。

(2) 大语言模型处理

根据查询出的结果，使用大模型以辅助处理，呈现出完整清晰的回答。

(3) 用户界面

最后，为了让用户能够方便地提问和查看答案，需要开发一个友好的用户界面。这可以是一个网页应用、移动应用或者与聊天机器人集成的界面。

1.5 论文组织架构

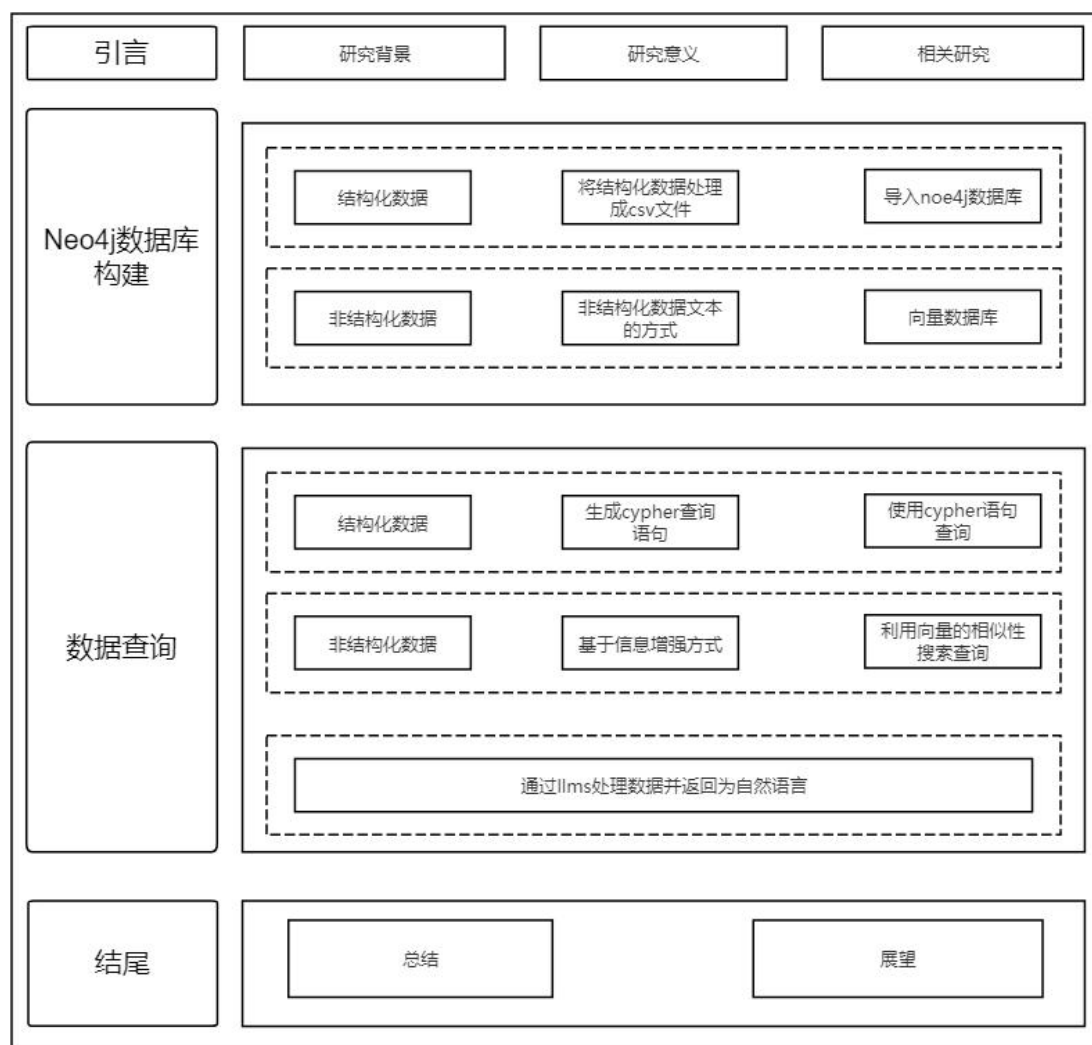


图 1-4 全文主要内容结构图

本文将按照以下结构展开，全面介绍基于知识图谱的足球知识问答系统的理论依据、设计和实现过程，如图 1-4 所示；

第1章 绪论

详细介绍研究背景和意义，阐述足球知识问答系统的发展现状和技术背景，综述国内外相关研究动态，并明确本文的研究内容、拟解决的问题、可行性分析及研究步骤，最后介绍论文的整体组织架构。

第2章 系统需求分析

深入探讨系统的研究内容、项目目标和范围定义，明确系统的功能性需求和非功能性需求。通过系统功能框图和系统流程图，详细说明系统的设计需求，为后续的设计和实现奠定基础。

第 3 章 基于知识图谱的足球知识问答系统的构建与推理

介绍系统的架构设计，包括数据收集与预处理、知识图谱的构建、自然语言处理的应用以及问答推理机制。具体内容包括如何从多种数据源收集和处理足球数据，如何利用 Neo4j 数据库构建知识图谱，如何通过自然语言处理技术理解用户问题，并生成和执行 Cypher 查询语句。

第 4 章 系统实现

描述系统的实现过程，包括 Neo4j 数据库的设计、基于本地大语言模型的处理、问答系统的开发和实现。本章还介绍了系统的各个主要功能模块，如聊天用户界面、自然语言接口和前后端分离的实现方法。

第 5 章 系统测试

系统的功能性测试和非功能性测试，详细描述测试方法和步骤，并展示测试用例和结果。通过对系统的功能和性能进行全面测试，验证系统的可靠性和有效性。

第 6 章 系统部署与应用

系统的部署环境与工具，包括服务器配置、服务端和客户端的部署工具，详细描述项目构建和部署的实现过程。本章还包括系统的实际应用和部署过程的说明。

1.6 本章小结

本章主要介绍了足球知识问答系统的研究背景和意义，综述了国内外相关研究动态，明确了本文的研究内容和拟解决的问题，进行了技术、经济和操作可行性分析，并介绍了研究步骤和论文组织架构。通过本章的内容，读者可以对足球知识问答系统的研究背景和发展现状有一个全面的了解，为后续章节的详细研究和系统实现奠定基础。

第 2 章 系统需求分析

2.1 研究内容

基于知识图谱的足球知识问答系统，是一组由 Docker Compose 协调的 Docker 容器组成，其中包括本地 LLMs 的管理工具（Ollama）、用于接地的数据库（Neo4j）和基于 LangChain 的应用程序。这些容器提供了一个支持代理应用（RAG 应用程序背后的理念是在查询时为 LLMs 提供额外的上下文，以便回答用户的问题）程序的开发环境，其中包含数据导入和响应生成用例。尝试在知识图谱中导入相关的足球信息，并检查底层基础信息的多样性如何影响用户界面中 LLM 生成的响应。

本项目采用 vue3 作为前端支撑，后端采用 python 的 fastApi 库，redis 作为信息的缓存处理，使用 langchain 连接大模型以辅助 Neo4j 的数据库进行信息的检索。

基于知识图谱的足球知识问答系统包括：

应用程序容器（Python 应用程序逻辑，使用 LangChain 进行协调，使用 Streamlit 进行用户界面）。

具有向量索引和图搜索功能的数据库容器（Neo4j）。

LLM 容器 Ollama。

前后端分离项目，前端采用 vue3，后端采用 python 的 fastApi 库

2.2 项目目标与范围定义

项目目标与范围定义将主要探讨和阐述项目的总体目标和预期结果，该部分还需要定义项目的边界和范围，即包括什么、不包括什么。下面这部分将介绍系统的工作目标以及核心功能、在实施过程中的注意事项等。

2.2.1 项目目标

构建一个完整的足球知识图谱，整合足球相关的数据，构建结构化的知识图谱，支持高效的查询和检索。并且实现智能问答功能，通过自然语言处理技术，实现对用户提出的足球相关问题的解析和回答。最终提升用户体验，提供一个简洁、友好

的用户界面，使用户能够方便快捷地获取所需的足球信息。

2.2.2 范围定义

基于知识图谱的足球知识问答系统的范围大致包括已下几个部分：

(1) 数据收集与处理：

从各种来源收集足球相关数据，并对数据进行清洗和结构化处理。

(2) 知识图谱构建：

利用 Neo4j 数据库构建足球知识图谱，定义实体和关系，确保数据的完整性和一致性。

(3) 问答系统开发：

基于 Python 和 LangChain 框架开发问答系统，包括前端界面和后端逻辑。

(4) 本地 LLM 集成：

使用 Ollama 项目中的开源大语言模型，支持自然语言处理和复杂问答功能。

2.3 功能性需求

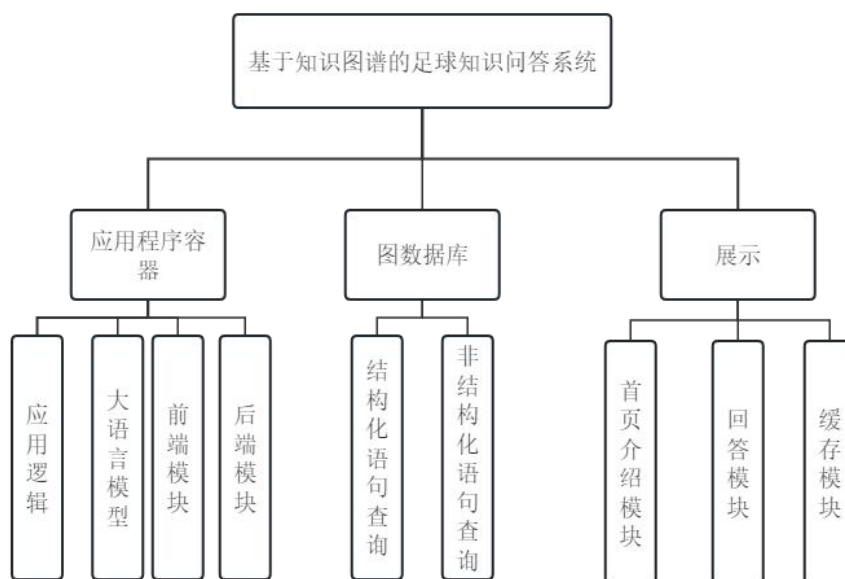


图 2-1 系统的功能框图

功能性需求将从系统功能模块的角度分析系统的实现的相关功能，以及用户能如何使用该系统的相关功能,如 2-1 所示。

系统功能框图（也称为系统框图或功能方框图）是系统整体功能设计图，用于清晰地表达系统的复杂结构关系和功能原理。它按照功能的从属关系画成图表，每个方框代表实际系统中的某个功能模块。系统的功能框图如图 2-1 所示。

系统流程图是一种图形化工具，用于表示系统的流程和工作机制。它通过一系列图形符号展示系统中各个步骤的顺序和相互关系，使复杂的流程清晰易懂。系统流程图（System Flowchart）是一种描述信息系统各部分如何交互的图表。它通过使用特定的图形符号（如矩形、菱形、箭头等）来表示系统中的各个步骤和流程，使得整个系统的工作机制一目了然。

本项目通过使用本地大语言模型和在线大语言模型相结合实现对问题的自然语言处理，并且通过是否开启 RAG 模式选项，实现定制化的实现功能，如图 2-2 所示。

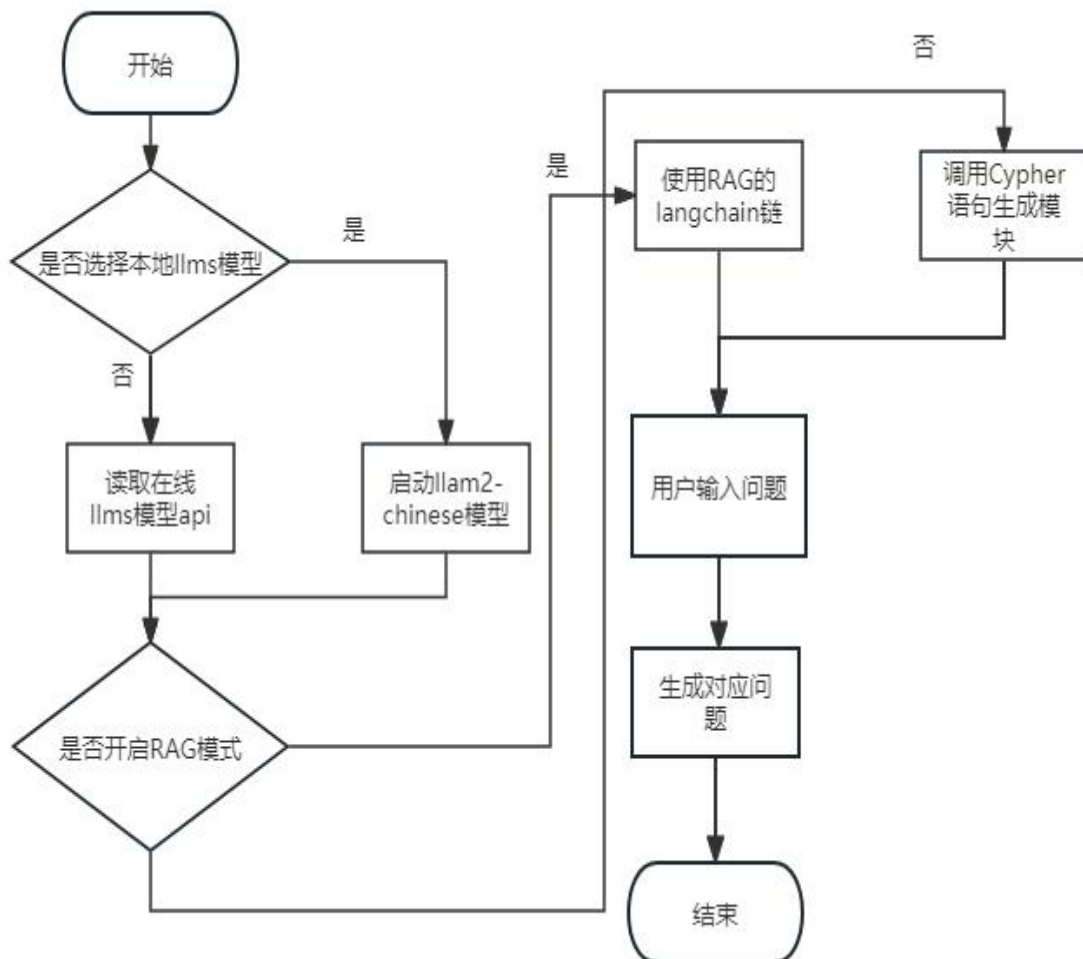


图 2-2 系统流程图

2.4 本章小结

本章详细分析了基于知识图谱的足球知识问答系统的需求，明确了项目的研究内容、目标与范围，系统功能性需求和非功能性需求。通过系统功能框图和流程图，展示了系统的复杂结构和工作机制。需求分析为系统设计和实现提供了明确的指导方向，奠定了坚实的基础。

第3章 基于知识图谱的足球知识问答系统的构建与推理

3.1 引言

本章将详细介绍基于知识图谱的足球知识问答系统的构建与推理过程。本文将从系统架构设计、知识图谱构建、数据处理与导入、自然语言处理技术的应用以及问答推理机制等方面进行详细描述。目标是为读者提供一个全面的理解，了解如何利用知识图谱和自然语言处理技术构建一个高效、准确的足球知识问答系统。

3.2 系统架构设计

系统架构设计是系统开发的基础，它决定了系统各个组件的组织方式和交互方式。基于知识图谱的足球知识问答系统的整体架构如图所示，主要包括以下几个模块：

(1) 数据收集与预处理模块：

负责从多个数据源收集足球相关数据，并进行数据清洗和结构化处理。

(2) 知识图谱构建模块：

利用 Neo4j 数据库构建足球知识图谱，定义实体和关系。

(3) 问答系统模块

基于 Python 和 LangChain 框架，实现自然语言处理、知识图谱查询和答案生成功能。

(4) 用户界面模块：

基于 Streamlit 框架，提供简洁、友好的用户交互界面。

3.3 知识图谱的构建

知识图谱构建是系统的核心部分，主要包括数据收集、实体识别、关系抽取和图谱存储等步骤。通过这些步骤，系统能够有效整合和关联各种足球数据，形成一个完整的知识网络，为用户提供精准、深入的分析和服。这种知识图谱能够帮助用户更好地理解复杂的足球信息使用户获得更全面的足球见解。

3.3.1 数据收集与预处理

数据来源包括足球赛事数据库（如 FIFA、UEFA 的数据）、球员和球队的官方网站、体育新闻和报道、专业足球统计网站（如 Transfermarkt、Opta）等。数据预处理包括数据清洗、格式转换和语义标注，确保数据的一致性和完整性，如图 3-1 所示。



图 3-1 节点类型

足球数据中有大量的结构化数据和非结构化数据，以下是对两种不同数据的介绍：

(1) 结构化数据

按照预定义格式组织的数据，这种数据类型高度组织，易于搜索和操作，适用于复杂的查询和分析。

例如球员可能包括球员 ID、姓名、位置、国籍等字段。

(2) 非结构化数据

指的是不符合固定格式或不容易被机器阅读的数据。这类数据在足球中很常见，

通常包含丰富的信息，但需要更复杂的处理方法来提取有用的数据。

文字评论：比赛的实时评论或赛后分析，通常包含了对球员表现和战术执行的描述。

社交媒体数据：球迷、球员或俱乐部的社交媒体帖子，反映了公众情绪和观点。

新闻文章：关于球队、球员或比赛的新闻报道。

3.3.2 实体识别与关系抽取

实体识别是从数据中识别出关键的足球实体，如球员、球队、比赛、联赛等。关系抽取是识别实体之间的关系，如“球员-效力于-球队”、“比赛-属于-联赛”等。使用自然语言处理技术和规则匹配方法，自动化地进行实体识别和关系抽取。

结构化数据设计

(1) 数据实体（节点类型）

在知识图谱中，定义了以下四种主要的实体类型，每种实体通过 MERGE 语句创建或确认存在，确保不会重复添加相同的实体：

Division：代表赛事的分区或级别，例如英超、西甲等。

Team：代表参赛的队伍，例如曼联、巴萨等。

Referee：代表裁判，记录裁判的姓名和执法记录。

Match：代表具体的比赛，包含比赛的日期、地点、参赛队伍和结果等信息。

(2) 实体关系

知识图谱中的实体通过以下关系进行关联，以准确描述足球领域中的各种互动和事件

(homeTeam)-[:PLAYS_IN]->(match)：表示主队参与了这场比赛。

(awayTeam)-[:PLAYS_IN]->(match)：表示客队参与了这场比赛。

(match)-[:HAS_HOME_TEAM]->(homeTeam)：表示这场比赛的主队是谁。

(match)-[:HAS_AWAY_TEAM]->(awayTeam)：表示这场比赛的客队是谁。

(match)-[:OFFICIATED_BY]->(referee)：表示这场比赛由哪位裁判主管。

(division)-[:CONTAINS]->(match)：表示某个分区包含了这场比赛。

通过上述关系实现了对英超比赛和球员的结构化保存，为接下来的使用大语言模型生成 Cypher 语句提供了数据的支持

3.3.3 图谱存储

使用 Neo4j 数据库存储构建好的知识图谱，定义节点类型（如球员、球队、比赛等）和关系类型（如效力、比赛关系等）。Neo4j 作为一款领先的图数据库，以其高效的关系查询能力和灵活的存储结构，特别适合处理复杂的知识图谱数据。本文将从结构化数据和非结构化数据两个角度，详细分析 Neo4j 在知识图谱存储中的应用。

Neo4j 部分创建语句，通过爬虫等手段爬取英超的数据，导入到 Neo4j 数据库，拥有 58700 条关系，10 个主要节点的图数据库，下面分别介绍主要的图结构。

(1) 结构化数据存储

对于结构化数据，Neo4j 提供了强大的图形数据模型和查询语言（Cypher），能够高效地存储和管理实体及其关系。结构化数据是指具有固定模式或格式的数据，例如球员信息、比赛结果、球队资料等。这些数据通常以表格或关系数据库的形式存在，适合用 Neo4j 的基础功能进行存储和查询，如图 3-2 图 3-4 所示。

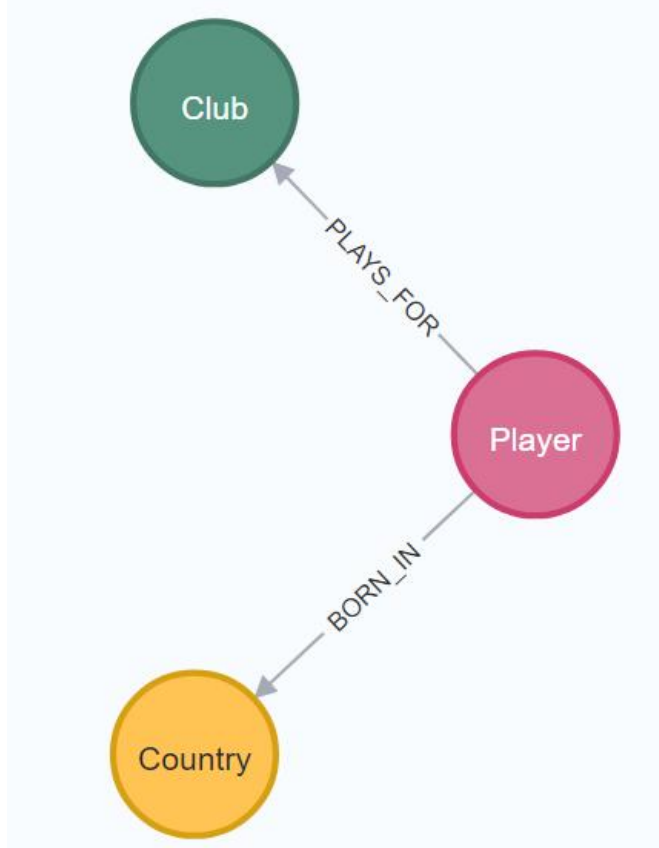


图 3-2 球员信息

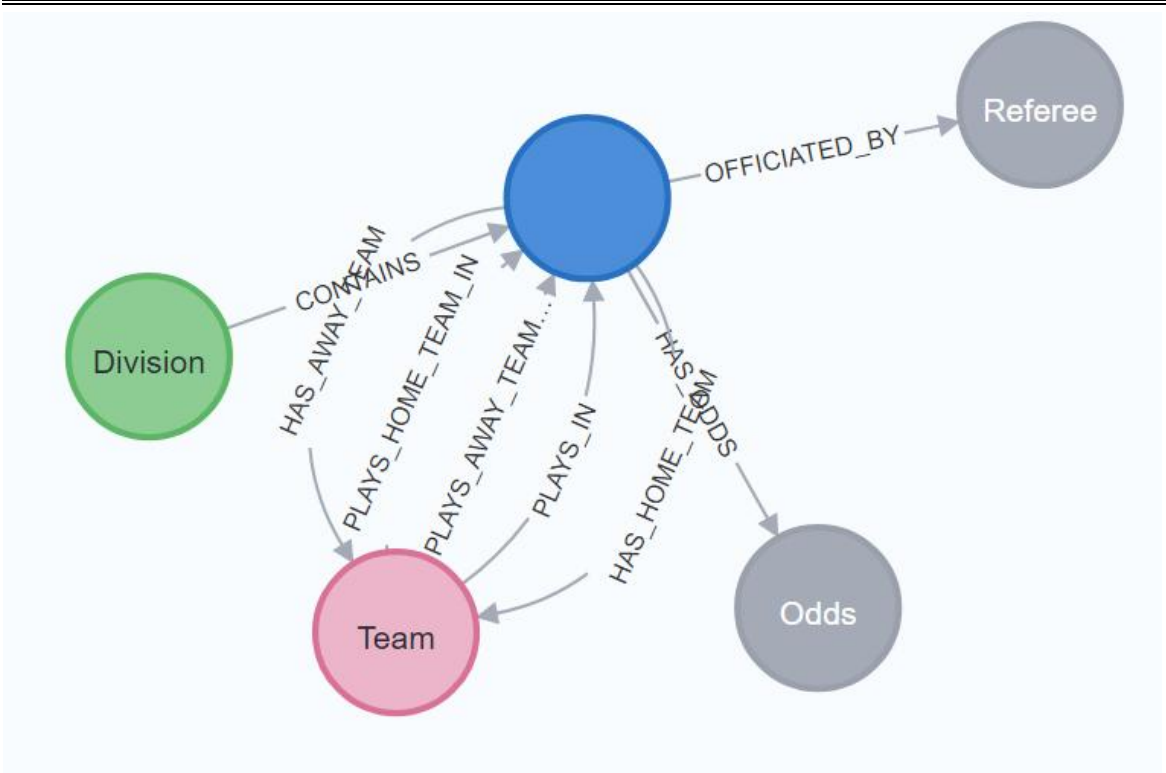


图 3-3 球队信息

| Key | Value |
|-----------|--|
| <id> | 6197 |
| id | "94c5dd94f155b062f683c64e1e163da0" |
| embedding | [0.01891249045729637, 0.023545116186141968, 0.009106782265007496, 0.008071010932326317, -0.0032139925751835108, 0.04980115592479706, -0.02085777744650... Show all |
| text | "==== 中锋 ==== 中锋 (Centre Forward, C F) 或称柱式中锋, 通常都比较高, 可以头锤和脚射门。他们通常倚靠头锤顶入传中球得分, 或使用身体和力量护球以制造攻门机会。通常由站桩式中锋或柔性站桩式中锋担任。 |

图 3-4 结构化数据

(2) 非结构化数据存储与查询

对于非结构化数据，如文本、图像、视频等，传统的图数据库处理能力有限。然而，通过 Neo4jVector，Neo4j 可以利用向量化技术进行相似性查询，从而高效地处理非结构化数据。向量化技术将非结构化数据转换为向量表示，使得在图数据库中能够进行高效的相似性搜索和匹配，如图 3-4 所示。

向量化数据存储：Neo4jVector 允许将非结构化数据向量化，并将这些向量存储在图数据库中。例如，对于一段比赛评论或球员的描述，可以使用预训练的语言模型（如 BERT、GPT）将文本转换为向量，并将这些向量与相关节点关联起来。

相似性查询：利用向量化数据，Neo4jVector 可以在图数据库中进行相似性查询。例如，可以查询与某个比赛描述最相似的其他比赛，或者找到与某个球员表现相似的其他球员。

3.4 自然语言处理

自然语言处理（NLP）是实现问答系统的关键技术。NLP 模块包括以下几个功能：

3.4.1 问题理解

通过 NLP 技术解析用户提出的问题，识别出问题中的关键实体和意图。使用预训练的语言模型（如 BERT、GPT）进行语义解析，提取出问题中的关键信息。

3.4.2 查询生成

根据解析结果，将用户问题转换为 Cypher 查询语句。Cypher 是 Neo4j 数据库的查询语言，能够高效地在知识图谱中进行查询。利用 LangChain 框架，将自然语言问题自动转换为结构化查询语句。

具体实现：

(1) 存储数据

Neo4j 一直并且非常擅长在 RAG 应用程序中存储和分析结构化信息。

(2) 提供信息

可以直接向 LLM 提供 schema 信息，让它仅基于图 schema 信息构建 Cypher

语句

(3) 首先语句

生成一个 Cypher 语句。

(4) 查询数据库

使用生成的 Cypher 语句来查询 Neo4j 数据库。

(5) 如果 Cypher 语法有效，则返回查询结果。

(6) 纠错出处理

如果存在 Cypher 语法错误，对 GPT-4 进行一次后续操作，提供在先前调用中构造的生成的 Cypher 语句，并包含来自 Neo4j 数据库的错误。GPT-4 非常擅长修复出现错误的 Cypher 语句，如图 3-5 所示。

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
cypher
MATCH (u:User {display_name: "A. L"})-[:ASKED]->(q:Question)
RETURN q ORDER BY q.view_count DESC LIMIT 1
```

图 3-5 生成的 cypher 语句

此外，Neo4j 添加了向量索引搜索，这使其更接近支持基于非结构化文本的 RAG 应用。

(1) 使用 LangChain 文档阅读器阅读维基百科文章

(2) 将文本分块

(3) 将文本存储到 Neo4j 中并使用新添加的向量索引对其进行索引

(4) 实施问答工作流程以支持 RAG 应用程序。

3.5 问答推理机制

问答推理机制是系统的核心功能，主要包括查询执行、答案生成和结果反馈。

3.5.1 查询执行

执行生成的 Cypher 查询语句是知识图谱问答系统中的关键步骤之一。在用户提出问题后，系统首先将自然语言问题转换为相应的 Cypher 查询语句。然后，这些查询语句将在 Neo4j 数据库中执行，以检索相关的知识图谱信息。Neo4j 以其高效的图

查询功能著称，能够快速处理复杂的图结构和关系，返回准确的查询结果，如图 3-6 所示。

```
query = "足球俱乐部是什么的"

results = neo4j_vector.similarity_search(query, k=5)
print(results[0].page_content)
```

足球俱乐部是足球职业化及专业化的一个标志，是足球员以足球谋生时所被聘用的机构。

== 历史 ==

第一家现代足球俱乐部是英国謝菲爾德足球俱樂部。全亞洲最歷史悠久而又運作至今的足球

== 经营 ==

足球俱乐部的收入主要依靠出售门票、经营广告、转让转播权、足球彩票、公司赞助、办各

== 参见 ==

足球俱乐部列表

图 3-6 使用使用文本嵌入和图形查询进行相似性搜索，在数据库中找到最相关的问题和答案。

查询语句生成

首先，系统利用自然语言处理（NLP）技术解析用户提出的问题，识别出问题中的关键实体和意图。例如，对于问题“梅西在哪个球队效力？”系统会识别出关键实体“梅西”和意图“效力于”，如图 3-7 所示。

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
cypher
MATCH (u:User {display_name: "A. L"})-[:ASKED]->(q:Question)
RETURN q ORDER BY q.view_count DESC LIMIT 1
```

图 3-7 生成的 cypher 语句

一旦生成了 Cypher 查询语句，系统将其提交给 Neo4j 数据库进行执行。Neo4j 数据库内部使用优化的图存储和索引机制，能够高效地处理和执行查询。具体执行过程如下：

查询解析与优化：Neo4j 首先解析接收到的 Cypher 查询语句，并生成查询计划。查询计划是查询执行的蓝图，包含了执行查询所需的步骤和顺序。Neo4j 会对查询计

划进行优化，确保查询能够以最小的资源消耗和最快的速度执行。

图遍历与数据检索：根据查询计划，Neo4j 在图数据库中进行图遍历和数据检索。对于每个匹配的节点和关系，Neo4j 会访问相关的存储块并提取所需的数据。由于 Neo4j 采用了高效的图存储结构和索引机制，图遍历和数据检索过程非常快速。

结果集生成：检索完成后，Neo4j 会将匹配的结果节点和关系组装成结果集。结果集可以是简单的属性列表，也可以是复杂的图结构，具体取决于查询语句的返回格式。

查询结果处理

查询执行完成后，Neo4j 会将结果集返回给问答系统。系统接收到查询结果后，需要对结果进行处理，以生成用户可理解的自然语言答案。例如，对于上述查询返回的结果“Paris Saint-Germain”，系统会将其转换为自然语言答案“梅西目前效力于巴黎圣日耳曼”。

3.5.2 答案生成

将查询结果转换为自然语言答案。利用预训练的语言模型，将结构化数据转化为用户可理解的自然语言回答。答案生成过程中，考虑上下文和用户的实际需求，确保答案的准确性和相关性。

3.5.3 结果反馈

通过用户界面将生成的答案反馈给用户。使用 Streamlit 框架，提供简洁、直观的交互界面，使用户能够方便地查看答案和提出新的问题，如图 3-7 图 3-8 所示。



图 3-7 问答系统（翻译为：查尔顿、切尔西、桑德兰、托特纳姆热刺和曼联是英超球队）



图 3-8 使用 llama-chinese 模型

3.5.3 实验与评估

为了验证系统的有效性和性能，本文设计了一系列实验，包括问答系统的准确性测试和响应时间评估。测试数据集包括常见的足球知识问题，如“梅西在哪个球队效力？”、“2022 年世界杯冠军是谁？”等。实验结果表明，系统在准确性和响应时间上表现良好，能够高效、准确地回答用户提出的足球问题。

3.7 本章小结

本章详细介绍了基于知识图谱的足球知识问答系统的构建与推理过程。从系统架构设计、知识图谱构建、自然语言处理到问答推理机制，各个环节相互协作，实现了一个高效、准确的问答系统。实验结果验证了系统的有效性，为后续的优化和应用提供了参考。下一章将进一步探讨系统的优化和扩展方法。

第 4 章 系统实现

4.1 基本服务介绍

目前为止基于知识图谱的足球知识问答系统，进行系统详细设计如表 1 所示，设计了以下服务：

- (1) bot 拥有经典的 LLM 聊天用户界面，用户可以提出问题并获得答案。
- (2) llm_chatbot 利用 GPT 模型将纯英文文本查询自动转换为 Cypher(知识图谱的查询语句)。
- (3) 知识图谱数据库
- (4) 前后端分离的基于知识图谱的足球知识问答系统。

表 1 基于知识图谱的足球知识问答系统主要功能模块介绍

| 名称 | 主要文件 | Urls | 描述 |
|--------------------|-----------------------|---|--|
| Bot | Bot.py | http://localhost:8501 | 拥有经典的 LLM 聊天用户界面，用户可以提出问题并获得答案 |
| llm_chatbot | llm_chatbot/ | http://localhost:8501 | 利用 GPT 模型将纯英文文本查询自动转换为 Cypher(知识图谱的查询语句) |
| Neo4j | Noe4j | http://localhost:7474 | 知识图谱数据库 |
| front-end 和 api | front-end 和 api.py | http://localhost:8505 | 前后端分离的基于知识图谱的足球知识问答系统。 |
| Bot | Bot.py | http://localhost:8501 | 拥有经典的 LLM 聊天用户界面，用户可以提出问题并获得答案 |
| llm_chatbot | llm_chatbot/ | http://localhost:8501 | 利用 GPT 模型将纯英文文本查询自动转换为 Cypher(知识图谱的查询语句) |

4.2 详细设计原理与成果介绍

4.2.1 Neo4j 数据库设计

通过爬虫等手段爬取英超的数据，导入到 Neo4j 数据库，拥有 15485 条关系，10 个主要节点的图数据库,下面分别介绍主要的图结构，如图 4-1 所示

(1) 球员信息（5000 条数据）

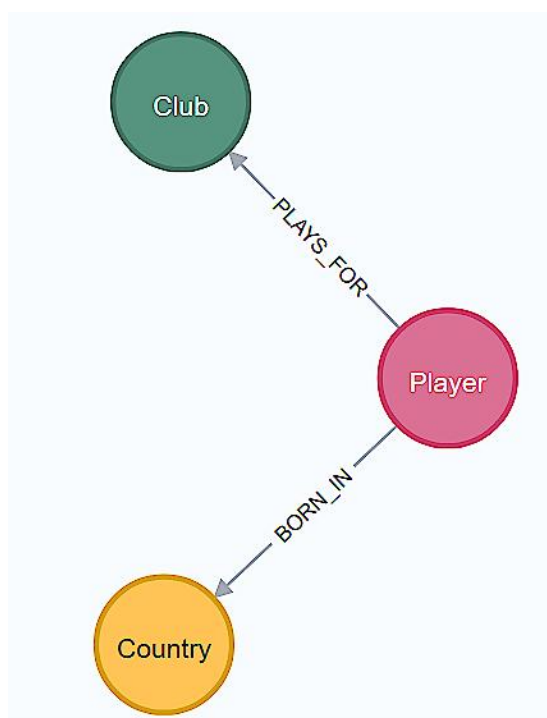


图 4-1 neo4j 部分图结构

如图 2 所示，主要节点有 3 个，分别为球员（player），所属国家（country），所在俱乐部（club），关系包括球员出生于（born_in）某个国家和球员效力(plays_for)于某个球队

(2) 比赛信息（2001 年到 2023 的英超数据）

如图 3 所示，主要节点包括赛区（Division），球队（Team），比赛（Match），裁判(Referee)

① 某个国家所有的所有球员

如图 4 某 country 的部分关系图,使用语句 `MATCH (n:Country) RETURN n LIMIT 25`

② 下面是数据库部分设计

节点类型 (Entities)

Division (赛区), Match (比赛), Team (球队), Referee (裁判)

关系类型 (Relationships)

Division-CONTAINS->Match (赛区包含比赛)

Match-HAS_HOME_TEAM->Team (比赛有主队)

Match-HAS_AWAY_TEAM->Team (比赛有客队)

Match-OFFICIATED_BY->Referee (比赛由裁判主持)

Team-PLAYS_IN->Division (球队在某赛区比赛)

通过上述关系实现了对英超比赛和球员的结构化保存, 为接下来的使用大语言模型生成 Cypher 语句提供了数据的支持

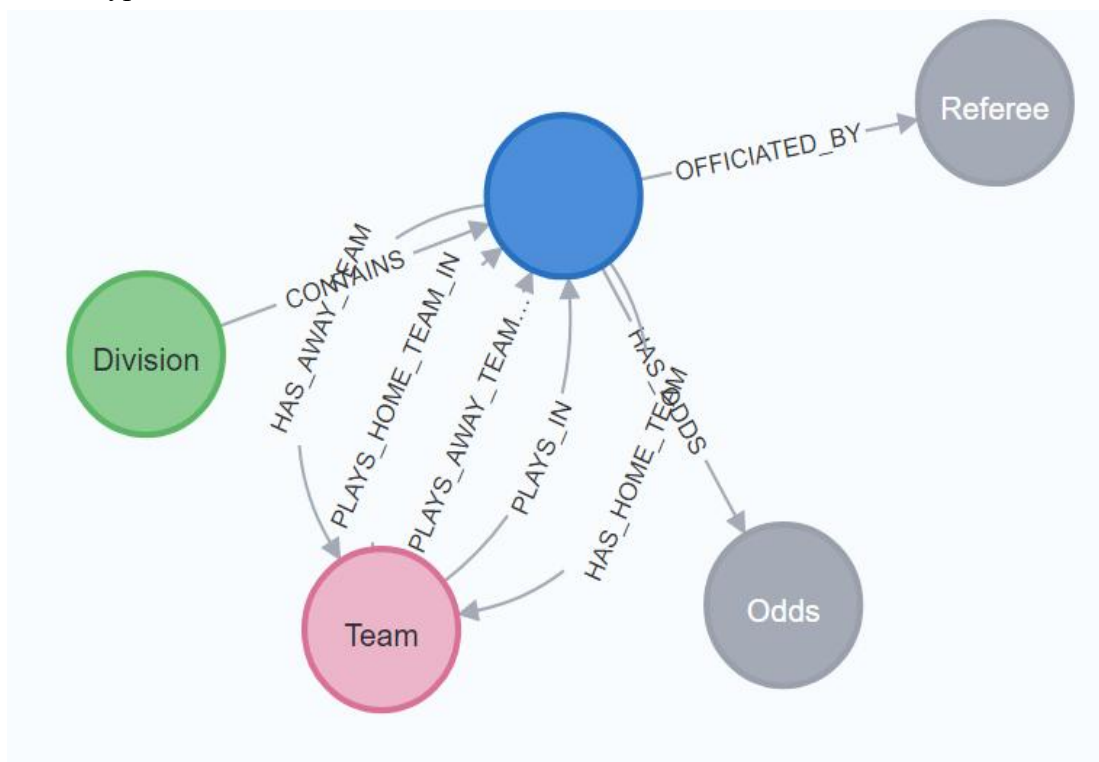


图 4-2 比赛信息

具体属性 (Attributes)

Match, Date (日期), Full Time Home Goals (全场主队进球), Full Time Away Goals (全场客队进球), Half Time Home Goals (半场主队进球), Half Time Away Goals (半场客队进球), Attendance (观众人数), Match statistics such as shots, fouls, cards (比赛统计: 射门、犯规、黄牌等), 如图 4-2 图 4-3 所示。

```
LOAD CSV WITH HEADERS FROM 'file:///datas/data1/CLEAN_FIFA23_official_data.txt' AS line FIELDTERMINATOR ','
MERGE (player:Player {
  id: line.ID,
  name: line.Name,
  age: toInteger(line.Age),
  datasetYear: 2023, // 这里直接设置为数据集的年份
  photo: line.Photo,
  nationality: line.Nationality,
  value: toFloat(line.ValueGBP),
  wage: toFloat(line.WageGBP),
  preferredFoot: line.PreferredFoot,
  position: line.Position,
  joined: line.Joined,
  contractValidUntil: toFloat(line.ContractValidUntil),
  height: toFloat(line.Height),
  weight: toFloat(line.Weight),
  yearJoined: toInteger(line.Year_Joined)
})
MERGE (club:Club {name: line.Club})
ON CREATE SET club.logo = line.ClubLogo
MERGE (player)-[:PLAYS_FOR {joined: line.Joined}]->(club)
MERGE (country:Country {name: line.Nationality})
ON CREATE SET country.flag = line.Flag
MERGE (player)-[:COMES_FROM]->(country)
```

图 4-3 部分 Cypher 语句

4.2.2 本地大语言模型

近来，开源 LLM 研究取得了显著进展。Llama2 和 Mistral 等模型显示出令人印象深刻的准确性和性能水平，使它成为商业模型的可行替代品。使用开放源码 LLMs 的一个显著好处是消除了对外部 LLM 提供商的依赖，同时保留了对数据流以及数据共享和存储方式的完全控制。Ollama 项目的维护者认识到了开源 LLMs 的机遇，他提供了一种无缝解决方案，可在您自己的基础架构甚至笔记本电脑上设置和运行本地 LLMs。

本项目使用了 llama2-chinese 模型，这个模型是基于 Meta Platform, Inc. 所发布的 Llama 2 Chat 开源模型来进行微调。根据 Meta，Llama 2 的训练数据达到了两万亿个 token，上下文长度也提升到 4096。对话上也是使用 100 万人类标记的数据微调。

由于 Llama 2 本身的中文对齐比较弱，开发者采用了中文指令集来进行微调，使其具备较强的中文对话能力。目前这个中文微调参数模型总共发布了 7B，13B 两种参数大小。

如图 4-4 所示，在 SuperCLUE 开放式多轮测评-十大能力成绩评估中，本文发现 Llama2 中文模型（FlagAlpha-Llama2-7b-Chat）在多数任务上效果总体效果还比较一般，多数能力的平均分离及格线都有比较大的差距。

| 排名 | 模型 | 胜和率 | 语义理解与抽取 | 闲聊 | 上下文对话 | 角色扮演 | 知识与百科 | 生成与创作 | 代码 | 逻辑与推理 |
|----|---------------------------|-------|---------|-------|-------|--------|-------|--------|-------|--------|
| - | gpt-4 | 94.64 | 80.00 | 97.30 | 93.18 | 100.00 | 87.76 | 100.00 | 97.92 | 100.00 |
| - | Baichuan-13B-Chat | 65.28 | 45.00 | 88.33 | 78.33 | 91.67 | 55.00 | 91.67 | 25.00 | 50.88 |
| - | ChatGLM2-6B | 36.50 | 33.33 | 38.33 | 36.67 | 41.67 | 20.00 | 40.00 | 21.67 | 55.00 |
| 1 | openbuddy_llama2_13b | 35.12 | 33.33 | 40.00 | 23.33 | 20.00 | 23.33 | 46.67 | 33.33 | 58.62 |
| 2 | Llama-2-13B-chat | 27.05 | 43.33 | 35.00 | 27.12 | 11.67 | 15.00 | 46.67 | 6.67 | 35.00 |
| 3 | FlagAlpha-Llama2-13b-Chat | 26.51 | 33.33 | 36.67 | 36.67 | 24.14 | 10.00 | 50.00 | 6.67 | 41.38 |
| 4 | Chinese-Alpaca-2-7B | 14.43 | 20.00 | 6.67 | 20.00 | 10.34 | 6.67 | 3.33 | 10.00 | 37.93 |
| 5 | firefly_llama2_13b | 12.54 | 16.67 | 6.90 | 3.57 | 3.33 | 0.00 | 6.67 | 16.67 | 46.67 |
| 6 | FlagAlpha-Llama2-7b-Chat | 12.50 | 17.24 | 20.69 | 16.67 | 3.33 | 6.67 | 13.33 | 3.33 | 26.67 |
| 7 | yayi_13b_llama2 | 8.78 | 16.67 | 0.00 | 10.00 | 3.33 | 3.45 | 3.33 | 10.34 | 20.00 |

图 4-4 2023 年 8 月 SuperCLUE 开放式多轮测评-Llama2 中文开源模型排行榜

但胜在拥有比较不错的中文回答，如图 4-4 所示，和 7B 大小的参数,实际效果还可以。



图 4-5 问答系统（翻译为：查尔顿、切尔西、考文垂、德比郡、利兹联、莱斯特城、利物浦、桑德兰、托特纳姆热刺和曼联是英超球队）

并且本项目通过构建一个 dockerfile 文件来拉取本地大语言模型，实现了本地大

语言模型的拉取的封装。



图 4-6 使用 llama-chinese 模型

4.2.3 Bot 项目

此外，Neo4j 几天前才添加了向量索引搜索，这使其更接近支持基于非结构化文本的 RAG 应用。

1. 使用 LangChain 文档阅读器阅读维基百科文章
2. 将文本分块
3. 将文本存储到 Neo4j 中并使用新添加的向量索引对其进行索引
4. 实施问答工作流程以支持 RAG 应用程序。

```
def configure_llm_only_chain(llm):
    # LLM only response
    template = """
    You are a helpful assistant that helps a support agent with answering programming questions.
    If you don't know the answer, just say that you don't know, you must not make up an answer.
    """
    system_message_prompt = SystemMessagePromptTemplate.from_template(template)
    human_template = "{question}"
    human_message_prompt = HumanMessagePromptTemplate.from_template(human_template)
    chat_prompt = ChatPromptTemplate.from_messages(
        [system_message_prompt, human_message_prompt]
    )

    def generate_llm_output(
        user_input: str, callbacks: List[Any], prompt=chat_prompt
    ) -> str:
        chain = prompt | llm
        answer = chain.invoke(
            {"question": user_input, config={"callbacks": callbacks}}
        ).content
        return {"answer": answer}

    return generate_llm_output
```

图 4-7 基于大语言模型的查询

拥有经典的 LLM 聊天用户界面，用户可以提出问题并获得答案。这里有一个名为 RAG 模式的开关，用户可以完全依赖 LLMs 训练有素的知识（RAG：已禁用），也可以使用功能更强的模式（RAG：已启用），即应用程序使用文本嵌入和图形查询进行相似性搜索，在数据库中找到最相关的问题和答案，如图 4-7 所示。

4.2.4 使用 LLMs 为 neo4j 提供自然语言接口

Neo4j 一直并且非常擅长在 RAG 应用程序中存储和分析结构化信息。

可以直接向 LLM 提供 schema 信息，让它仅基于图 schema 信息构建 Cypher 语句，run 函数首先生成一个 Cypher 语句。然后，使用生成的 Cypher 语句来查询 Neo4j 数据库。如果 Cypher 语法有效，则返回查询结果。但是，假设存在 Cypher 语法错误。在这种情况下，本文对 GPT-4 进行一次后续操作，提供在先前调用中构造的生成的 Cypher 语句，并包含来自 Neo4j 数据库的错误。GPT-4 非常擅长修复出现错误的 Cypher 语句，如图 4-8 所示。

```
query = "足球俱乐部是什么的"

results = neo4j_vector.similarity_search(query, k=5)
print(results[0].page_content)
```

足球俱乐部是足球职业化及专业化的一个标志，是足球队员以足球谋生时所被聘用的机构。

== 历史 ==

第一家现代足球俱乐部是英国谢菲尔德足球俱乐部。全亚洲最历史悠久而又运作至今的足球

== 经营 ==

足球俱乐部的收入主要依靠出售门票、经营广告、转让转播权、足球彩票、公司赞助、办各

== 参见 ==

足球俱乐部列表

图 4-8 通过相似性搜索 RAG 查询问题

使用 langchain 连接 neo4j 数据库，连接到 Neo4j 实例，并在初始化时获取模式信息。然后，图模式信息可用作 GPT-4 模型的输入。

图形模式以字符串格式存储

f"This is the schema representation of the Neo4j database.

Node properties are the following:

{node_props}

Relationship properties are the following:

{rel_props}

Relationship point from source to target nodes

{rels}

Make sure to respect relationship types and directions"

接下来做一些提示工程，为 GPT-4 模型创建一个系统提示，定义一个生成 Cypher 语句的函数如图 4-9 所示。该函数将接受特定的参数，如节点类型、属性条件等，自动构建并返回有效的 Cypher 查询语句，简化数据库流程，提高开发效率。

```
def construct_cypher(self, question, history=None):
    messages = [
        {"role": "system", "content": self.get_system_message()},
        {"role": "user", "content": question},
    ]
    # Used for Cypher healing flows
    if history:
        messages.extend(history)

    completions = openai.ChatCompletion.create(
        model="gpt-4",
        temperature=0.0,
        max_tokens=1000,
        messages=messages
    )
    return completions.choices[0].message.content
```

图 4-9 生成 Cypher 语句

生成 Cypher 语句，如图 4-11 所示

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
cypher
MATCH (u:User {display_name: "A. L"})-[:ASKED]->(q:Question)
RETURN q ORDER BY q.view_count DESC LIMIT 1
```

图 4-11 使用 LLMs 生成 Cypher 语句

4.2.5 前后端分离项目

本项目旨在开发一个具有与现有应用 bot 相同功能的新应用程序。不同于原有的架构，新的应用程序将采用当前最先进的前端和后端分离技术，以提升性能、可维护性和用户体验。通过采用这种现代化的方法，项目团队预期能够提供一个更灵活、易于扩展的解决方案，同时降低未来维护的复杂性和成本，如图 4-12 所示。

前端技术：使用 Vite、vue3 和 Tailwind CSS。

Vite：作为现代前端开发环境和构建工具，Vite 提供了极快的冷启动、即时热模块更新和真正的按需编译，从而极大地提高开发效率。

vue3：Vue.js 是一个开源的 JavaScript 框架，主要用于构建单页应用（SPA）和用户界面。与其他大型框架不同，Vue.js 被设计为可以逐步采用的，因此你可以根据项目需求，从简单的库使用扩展到更复杂的全功能框架。

Tailwind CSS：一个实用主义的 CSS 框架，用于快速定制设计，它允许开发者通过功能类直接在标记中构建设计，从而加快开发速度并减少样式表的大小。

前后端分离架构有助于提高开发效率、增强系统的可维护性和扩展性，使团队能够更灵活地扩展和维护系统。前后端分离架构将前端和后端分离开发与部署，前端负责用户界面和交互，后端负责业务逻辑和数据处理。

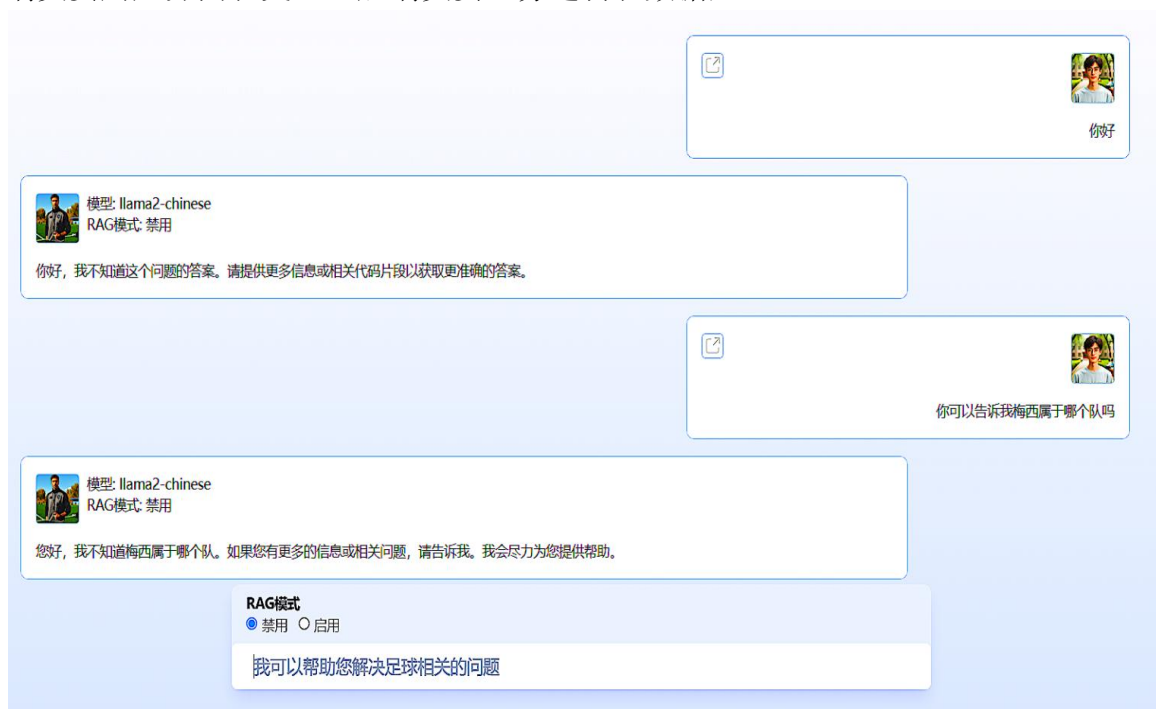


图 4-12 前端展示

4.6 本章小结

本章详细介绍了基于知识图谱的足球知识问答系统的实现过程。包括 Neo4j 数据库设计、数据导入、自然语言处理模块的实现以及前后端分离架构的开发。描述了系统各主要功能模块，如聊天用户界面和自然语言接口的实现方法。通过本章的内容，展示了系统从设计到实际实现的过程，为后续的系统测试和部署提供了扎实的基础。

第 5 章 前后端分离项目

本章将详细介绍基于前后端分离架构的足球知识问答系统。本系统前端使用 Vue3 框架构建，后端采用 FastAPI 实现，并使用 Redis 对问答结果进行缓存处理。系统主要分为三个模块：首页展示界面、问答界面和 RAG 模式支持。

1.1 系统架构

前后端分离架构将前端和后端分离开发与部署，前端负责用户界面和交互，后端负责业务逻辑和数据处理。这样的架构有助于提高开发效率、增强系统的可维护性和扩展性。

5.1.1 前端架构

前端使用 Vue3 框架构建，主要负责用户界面的呈现和交互逻辑。Vue3 以其高效、灵活和易用的特点，适用于构建现代化的单页应用程序（SPA）。

- (1) Vue3 框架：用于构建响应式用户界面
- (2) Vue Router：用于管理前端路由，实现单页应用导航
- (3) Vuex：用于状态管理，维护全局状态
- (4) Axios：用于与后端 API 进行通信

5.1.2 后端架构

后端采用 FastAPI 框架实现，FastAPI 以其高性能和易用性在构建现代 Web API 时非常适用。后端主要负责处理客户端请求、执行业务逻辑以及与数据库进行交互。

- (1) FastAPI 框架：用于构建高性能 Web API
- (2) SQLAlchemy：用于 ORM（对象关系映射），与数据库进行交互
- (3) Redis：用于缓存问答结果，提高响应速度和系统性能

本项目使用 Swagger，它可以帮助开发人员更快、更简单地设计、构建、文档化和测试 RESTful API。Swagger 可以自动生成交互式 API 文档、客户端 SDK、服务

器 stub 代码等，如图 5-1 所示。

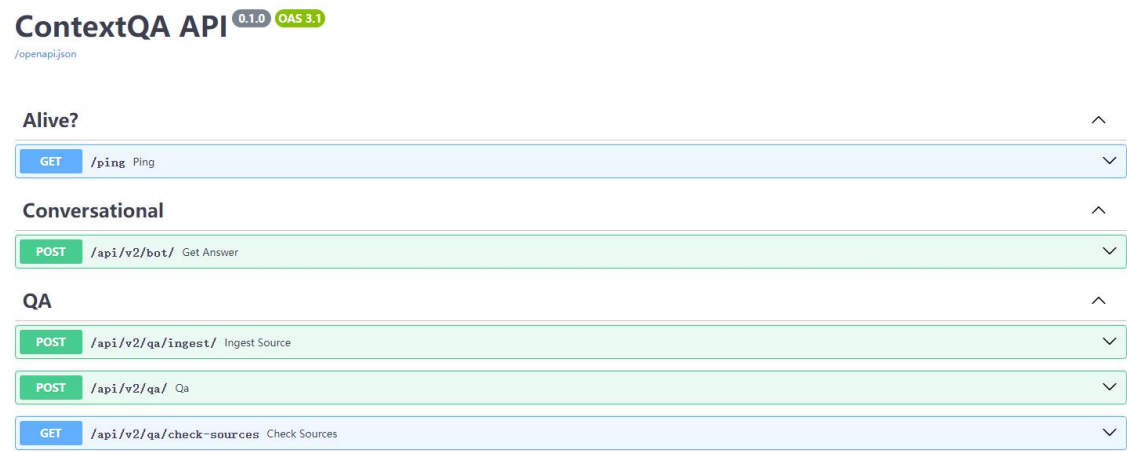


图 5-1 后端接口介绍

5.1.3 获得回答接口

发送一个 post 请求，再后端处理，返回回复的功能，restful 风格的接口，如图 5-2 所示，如图 5-3 所示

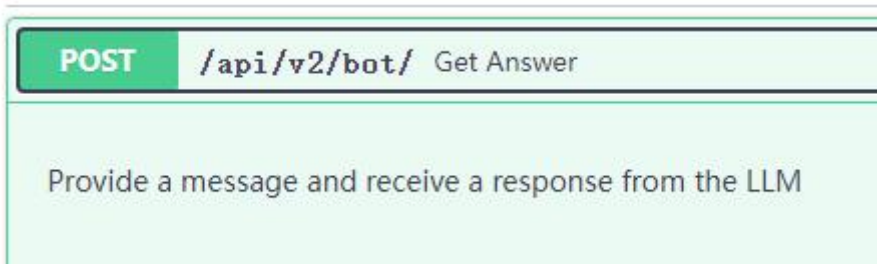


图 5-2 发送消息并且收到回复接口



图 5-3 请求体(样例)

响应体分为调用成果（200），调用失败（422），成果会返回一个字符串。失败则会返回一个通用响应，提示调用接口失败，通一返回 `application/json` 格式，如图 5-4 所示。

The screenshot displays a 'Responses' section with two entries:

| Code | Description |
|------|---------------------|
| 200 | Successful Response |
| 422 | Validation Error |

For the 200 response, the 'Media type' is set to `application/json`, and the 'Example Value' is `"string"`.

For the 422 response, the 'Media type' is also set to `application/json`, and the 'Example Value' is a JSON object:

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

图 5-4 响应体

5.1.4 qa 接口

发送一个 post 请求，再后端处理，返回回复的功能，restful 风格的接口，如图 5-5 所示。

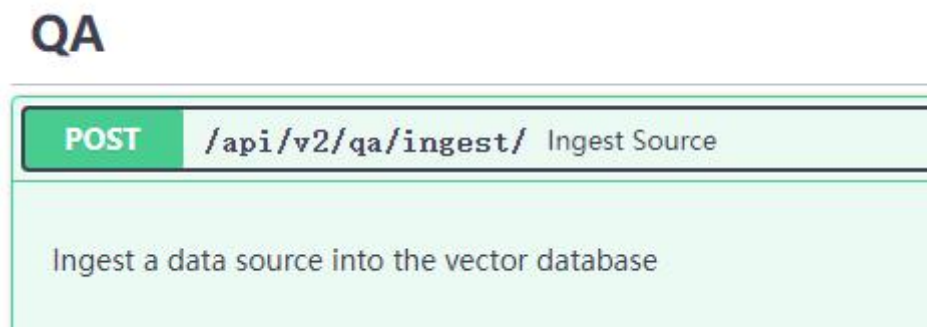


图 5-5qa 接口

5.1.3 缓存机制

下为了提升系统性能和响应速度，系统使用 Redis 对问答结果进行缓存处理。缓存机制可以减少对数据库的直接查询次数，降低系统负载。

Redis 缓存：存储问答结果，根据问题的唯一标识符（如哈希值）进行快速查询和返回

5.2 模块设计

5.2.1 首页展示界面

首页展示界面是用户与系统交互的起点，它供基本信息，激发用户的兴趣，引导他们深入了解和使用系统。以下是对首页展示界面内容的扩充建议：

项目简介：简要介绍系统的核心目的，提供一个智能化的足球知识问答平台，帮助用户解决各种足球相关的问题。

功能亮点：详细列出系统的关键功能，如实时问答、多语言支持、联网搜索、定制化搜索等。

本项目在问答界面提供实时反馈和操作提示，确保用户在任何操作阶段都能得到清晰指导。界面布局灵活，自适应不同设备和屏幕尺寸，保证用户在各种环境下都能获得一致的优质体验。



图 5-6 首页展示界面

5.2.2 问答界面

问答界面是用户与系统进行互动的主要场所，用户可以通过此界面输入问题并获取答案。界面设计简洁直观，方便用户使用问答界面采用清晰布局，支持文本输入，确保用户快速提问。个性化推荐帮助用户发现相关问题，增强互动体验。

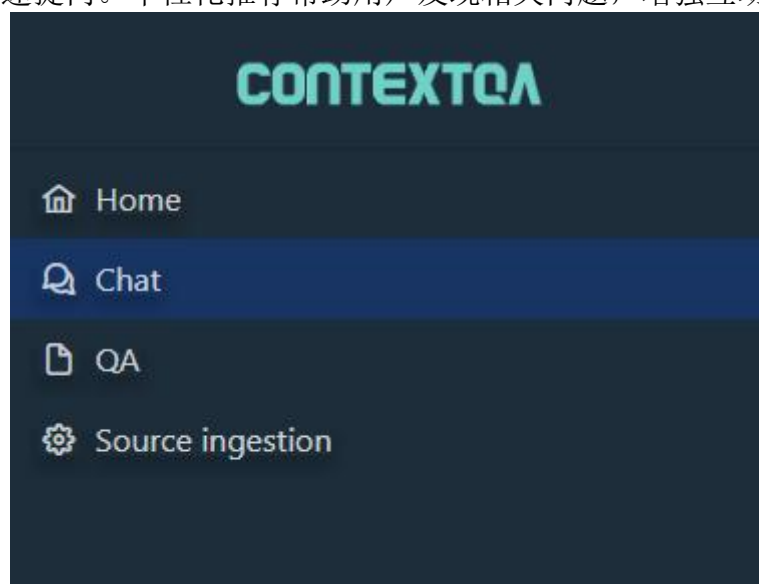


图 5-7 目录界面

侧边栏：包括首页，聊天，问答，上传文件等选项，如图 5-7 所示

输入框：用户输入问题的区域，如图 5-8 所示

提交按钮：提交问题进行查询

答案展示区域：显示系统返回的答案，如图 5-9 所示。

RAG 模式开关：支持用户开启或关闭 RAG 模式，开启后可以对非结构化数据进行查询

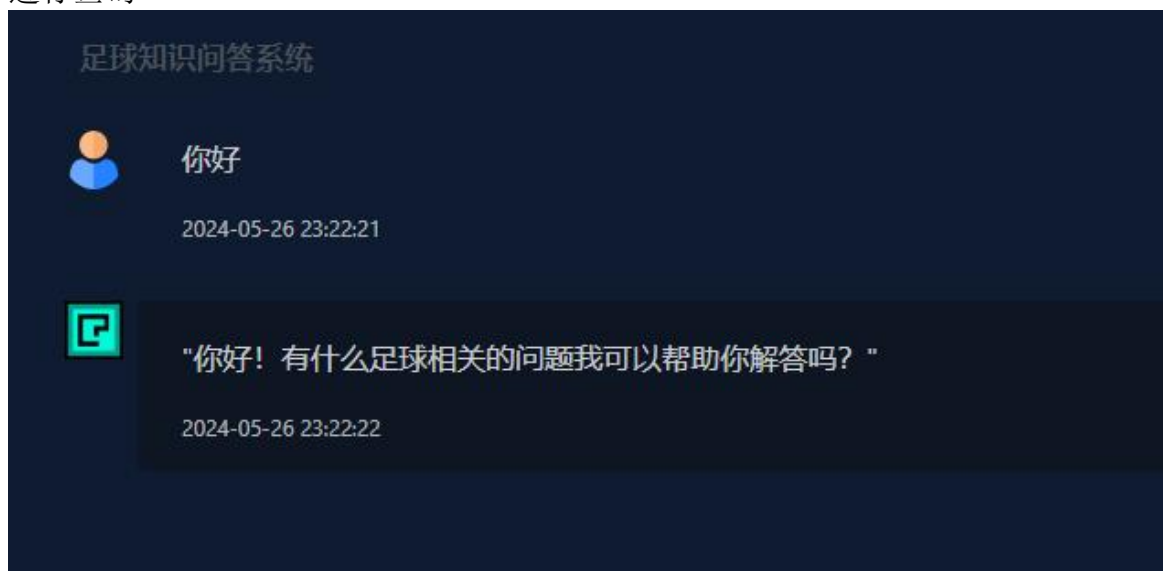


图 5-9 问答模块



图 5-8 输入框

5.2.3 RAG 模式支持

RAG（Retrieval-Augmented Generation）模式支持对足球信息的非结构化数据进行查询。启用 RAG 模式后，系统会结合结构化和非结构化数据，提供更全面和准确的回答，如图 5-10 所示。



图 5-10 开启 RAG 模式

开启 RAG 模式后，弹出窗口简洁明了地概述了增强功能，如更深入的分析、个

性化推荐和历史互动记忆，使用户能够更全面地利用系统资源，提升交互质量。如图 5-11 所示

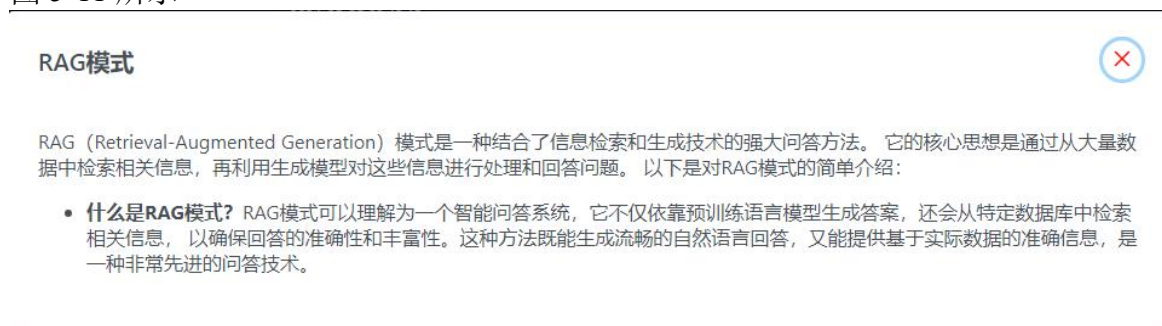


图 5-11 RAG 模式弹窗

非结构化数据查询：利用大语言模型对文本、图像等非结构化数据进行处理和查询

例如当询问足球一个队有多少人的时候，系统将会通过向量的相似性搜索，搜索出最符合要求的前 5 个备选答案，再通过 llm 对数据进行加工处理，得到正确的回复，如图 5-12 所示。



图 5-12 RAG 查询（非结构化数据查询）

增强回答准确性：结合知识图谱中的结构化数据和非结构化数据，生成更准确的回答

结构化数据查询，通过通义千问等大模型，生成 Cypher 查询语句实现进行高效的辅助查询功能，如图 5-13 RAG 查询（结构化数据查询）所示。当提问英超有哪些队伍的时候能够通过查询 Neo4j 数据库，清晰准确的提供出结果，



图 5-13 RAG 查询（结构化数据查询）

5.2.3 上传界面

足球小助手支持用户上传与足球相关的文件，如比赛数据、球员资料等。系统结合知识图谱进行深度分析，提供定制化的问答服务，帮助用户更好地理解和掌握足球信息。如图 5-14 所示，上传功能界面直观、操作简单。用户只需几个步骤即可上传文件，系统便会自动解析并生成个性化的分析报告和见解。这一功能确保了用户能够轻松获取到精准和个性化的足球信息。

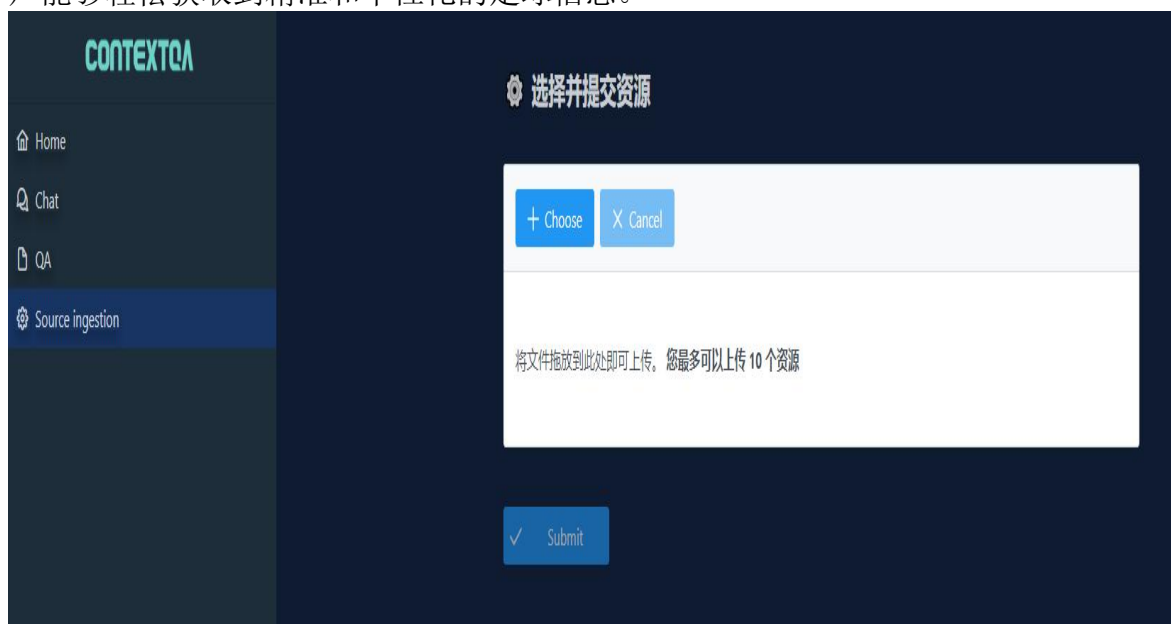


图 5-14 上传文件界面

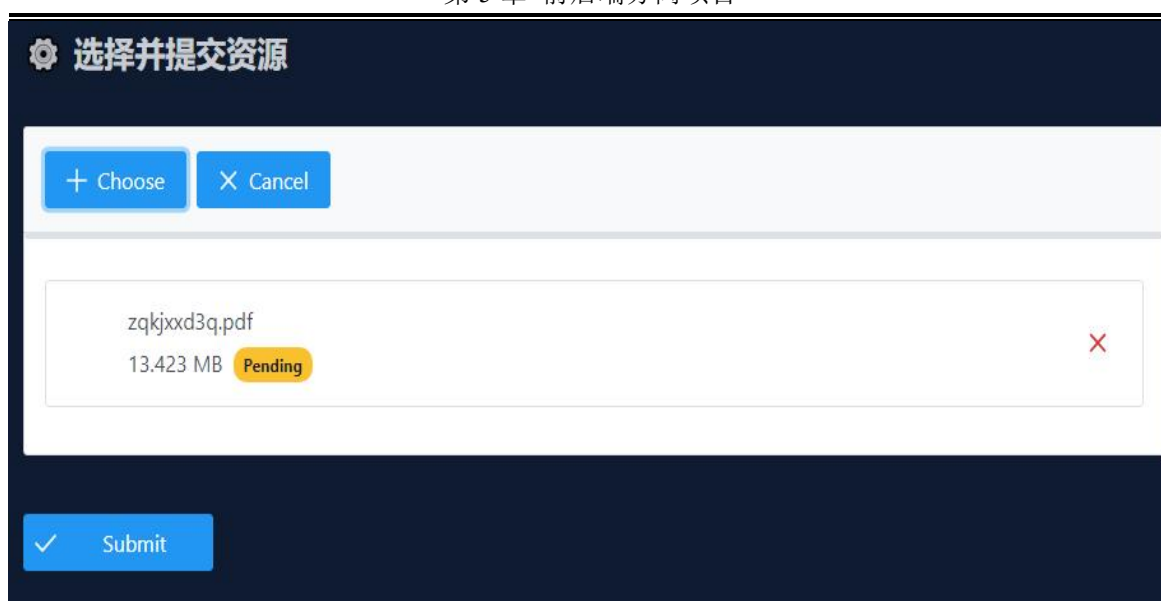


图 5-15 上传了一份 pdf 文件

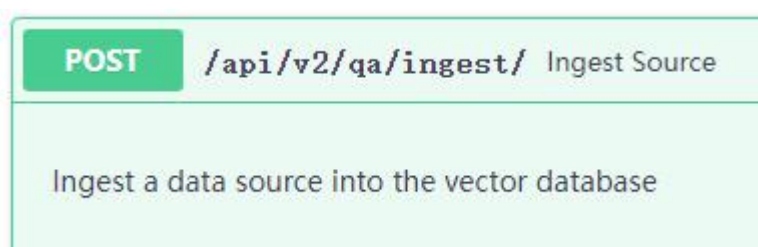


图 5-16 上传文件对应的后端接口

上传文件功能是足球小助手系统中的一项重要特性，它允许用户将本地的足球相关数据文件上传至系统。

系统支持多种文件格式，包括但不限于 TXT、PDF、Word 文档等，满足不同用户的需求。智能解析：上传后，系统将智能解析文件内容，提取关键信息，如球员表现、比赛结果、统计数据等。

系统利用知识图谱技术，将上传的数据与已有的足球知识库相结合，提供更深入的分析和更准确的问答。

用户界面：上传界面友好，提供清晰的指示和反馈，引导用户完成上传过程。

持续更新：系统会根据用户上传的内容持续更新知识库，以提供最新的足球信息和趋势分析。

发送一个 post 请求，上传一个文件，经过后端对文件进行分块处理，存储到 neo4j 数据库，提供一个 restful 风格的接口，如图 5-16 所示，如图 5-17 所示

| Code | Description |
|------|---|
| 200 | Successful Response |
| | Media type application/json |
| | Controls Accept header. |
| | Example Value Schema |
| | <pre>{ "completed": 0, "skipped_files": ["string"]}</pre> |
| 422 | Validation Error |
| | Media type application/json |
| | Example Value Schema |
| | <pre>{ "detail": [{ "loc": ["string", 0], "msg": "string", "type": "string" }]}</pre> |

图 5-17 后端接口的成果和失败的 json 格式

5.4 本章小结

本章详细介绍了前后端分离架构的足球知识问答系统的设计与实现，包括系统架构、模块设计和系统部署。前端采用 Vue3 框架，这是一款现代化的前端开发工具，以其组件化、响应式设计和易于维护的特点，为用户提供了流畅且直观的交互体验。

后端则选用了 FastAPI，这是一个高性能的异步 Web 框架，它支持 Python 3.6+，能够快速开发出高效、功能丰富的 API，并结合 Redis 缓存机制，系统能够提供高效、准确的问答服务，满足用户对足球知识的查询需求。

第 6 章 系统部署与应用

6.1 系统部署环境与工具

系统的部署环境和工具是确保系统稳定运行和高效服务的关键。本节将详细介绍服务器信息、服务端和客户端部署工具。

6.1.1 服务器相关信息

部署足球知识问答系统需要选择合适的服务器，服务器的配置直接影响系统的性能和稳定性。以下是服务器相关信息的详细介绍：

服务器类型：云服务器或物理服务器

操作系统：Linux 发行版

CPU：多核处理器，建议 8 核以上

内存：至少 16GB，建议 32GB 以上

存储：SSD 硬盘，建议 500GB 以上

网络带宽：高带宽，支持低延迟和高并发访问

6.1.2 服务端部署工具

服务端部署工具用于管理和部署服务端的各个组件，确保系统能够高效、可靠地运行。主要的服务端部署工具包括：

FastAPI 和 Vue 3 是两种流行的框架，分别用于后端和前端开发，结合使用可以创建高性能、现代化的 Web 应用程序。

FastAPI 是一个用于构建 API 的现代、快速（高性能）的 Web 框架，基于 Python 3.7+ 的类型提示。它的主要特点包括：

高性能：得益于 Starlette 和 Pydantic，FastAPI 可以处理大规模的并发请求，非常适合需要高性能的 API 应用。

快速开发：由于其直观的设计和强类型系统，开发速度显著提升，同时减少了调试时间。

自动生成文档：FastAPI 自动生成 OpenAPI 和 JSON Schema 文档，使 API 更易于使用和理解。

类型安全：利用 Python 的类型提示，确保代码的健壮性和可维护性。

Vue 3 是一个用于构建用户界面的渐进式 JavaScript 框架。与其前身 Vue 2 相比，Vue 3 带来了显著的性能提升和新特性。其主要特点包括：

组合式 API：提供更灵活和可组合的代码结构，增强代码复用性和可读性。

性能优化：更快的虚拟 DOM 和编译器优化，使 Vue 3 在大型应用中表现出色。

更好的 TypeScript 支持：改进的类型推断和支持，使得与 TypeScript 的集成更加顺畅。

Vue 3 CLI：提供了一套强大的脚手架工具，简化了项目的创建和管理。

Docker：用于容器化应用，简化应用的部署和管理

Docker Compose：用于定义和运行多容器 Docker 应用

Neo4j：图数据库，用于存储和查询知识图谱数据

Ollama：本地大语言模型管理工具

Git：版本控制工具，用于代码管理和协作开发

FastApi: FastAPI 是一个用于构建 API 的现代、快速（高性能）的 web 框架，使用 Python 并基于标准的 Python 类型提示

6.1.3 客户端部署工具

客户端部署工具用于管理和部署前端应用，使用户能够通过浏览器访问系统。主要的客户端部署工具包括：

Node.js：JavaScript 运行时，用于前端开发和构建

Vite：现代前端开发工具，用于快速构建和热更新

Vue：前端框架，用于构建用户界面

Tailwind CSS：实用主义 CSS 框架，用于快速定制设计

6.2 构建与部署的实现过程

本节详细描述了项目的构建与部署过程，包括项目构建、服务端镜像与容器的创建、以及 Nginx 服务的配置，前端的所需的文件如图 6-1 所示。

6.2.1 项目构建

后端使用 python 语言进行编写，目前为止基于知识图谱的足球知识问答系统，设计了以下服务

| 名称 | 修改日期 | 类型 | 大小 |
|-------------------|-----------------|--------------------|--------|
| .idea | 2024/5/24 18:00 | 文件夹 | |
| dist | 2024/5/24 12:10 | 文件夹 | |
| node_modules | 2024/5/24 12:29 | 文件夹 | |
| public | 2024/5/24 9:37 | 文件夹 | |
| src | 2024/5/24 9:37 | 文件夹 | |
| .browserslistrc | 2024/5/24 9:37 | BROWSERSLIST... | 1 KB |
| .dockerignore | 2024/5/24 9:37 | DOCKERIGNOR... | 1 KB |
| .editorconfig | 2024/5/24 9:37 | Editor Config 源... | 1 KB |
| .env.development | 2024/5/25 8:32 | DEVELOPMENT ... | 1 KB |
| .env.production | 2024/5/24 9:37 | PRODUCTION ... | 1 KB |
| .eslintrc.js | 2024/5/24 9:37 | JavaScript 源文件 | 1 KB |
| babel.config.js | 2024/5/24 9:37 | JavaScript 源文件 | 1 KB |
| Dockerfile | 2024/5/24 9:37 | 文件 | 1 KB |
| jsconfig.json | 2024/5/24 9:37 | JSON 源文件 | 1 KB |
| package.json | 2024/5/24 12:25 | JSON 源文件 | 1 KB |
| package-lock.json | 2024/5/24 12:29 | JSON 源文件 | 859 KB |
| README.md | 2024/5/24 9:37 | Markdown File | 1 KB |
| vue.config.js | 2024/5/24 9:37 | JavaScript 源文件 | 1 KB |

图 6-1 前端目录

- (1) bot 拥有经典的 LLM 聊天用户界面，用户可以提出问题并获得答案。
- (2) llm_chatbot 利用 GPT 模型将纯英文文本查询自动转换为 Cypher(知识图谱的查询语句)。
- (3) 知识图谱数据库
- (4) 前后端分离的基于知识图谱的足球知识问答系统。

6.2.2 Docker 的配置

Docker 是一种开源的容器化平台，旨在简化应用程序的开发、部署和运行。它通过将应用程序及其依赖项打包在一个可移植的容器中，确保在不同环境中的一致性和可靠性。Docker 容器是轻量级的，不需要额外的操作系统开销，能够在任何地方运行——从开发人员的笔记本电脑到生产环境的服务器。

Docker 的核心组件包括：

Docker Engine：一个轻量级的运行时和工具，用于创建和运行容器。

Docker Images：只读的模板，用于创建容器，包含应用程序及其所有依赖项。

Docker Containers：基于镜像运行的实例，包含应用程序及其环境。

Docker Hub：一个公共注册表，用于存储和分发 Docker 镜像。

通过 Docker，开发者可以利用容器技术，实现更快的应用交付周期、更高的资源利用率和更简化的运维流程，广泛应用于微服务架构、持续集成/持续部署（CI/CD）、云计算等领域。

本项目利用了 Docker 和 Docker Compose 来构建和管理一个由多个服务组成的复杂应用。这种方法允许单独配置和管理各个组件，实现服务的解耦合和灵活部署；如图 6-2 所示。

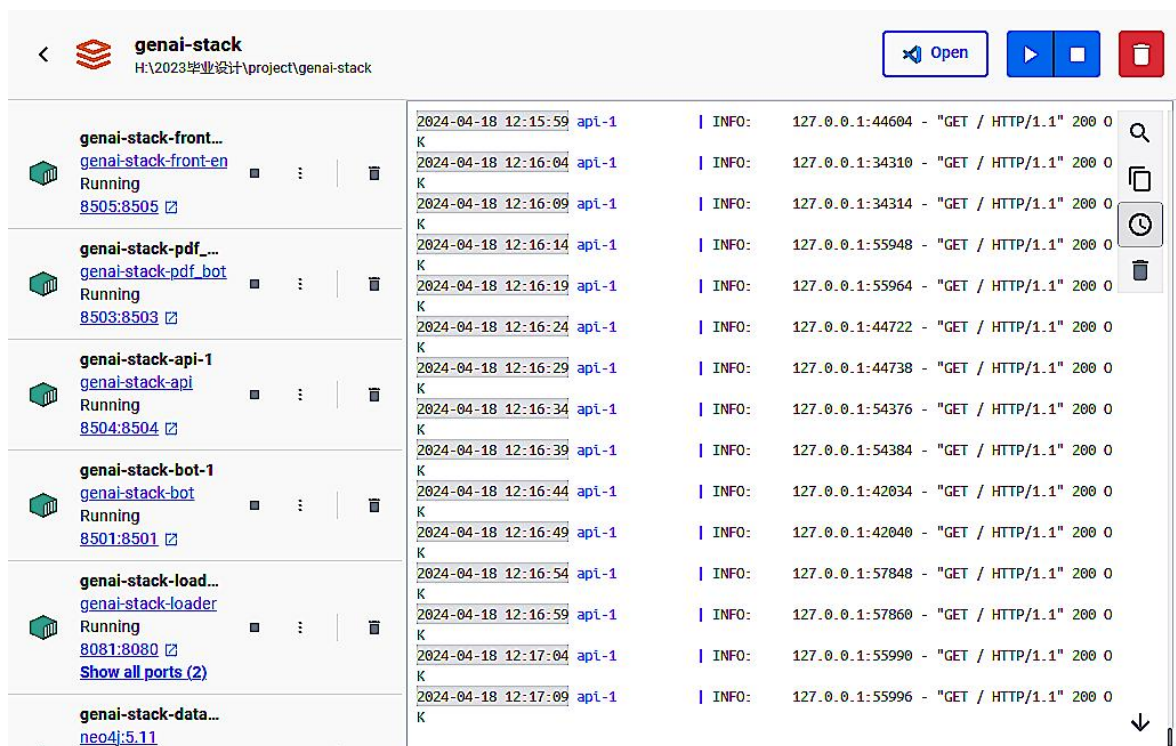


图 6-2 docker desktop 运行图

(1) 服务定义：

定义了多个服务，每个服务都可以单独构建和配置。这包括基本的应用服务（如 API 服务、前端服务）、数据库服务（如 Neo4j）、以及特定的功能服务（如模型拉取服务）。

(2) 环境配置：

通过环境变量，为每个服务提供了必要的配置，这些变量可以在运行时进行调整。这包括数据库连接信息、API 密钥、服务 URL 等。

(3) 网络配置：

所有服务都连接到同一个 Docker 内部网络 (net)，这样它就可以互相发现和通信。

(4) 依赖管理：

使用 `depends_on` 属性来管理服务间的依赖关系，确保例如数据库和模型拉取服务在其他依赖它的服务之前启动并且处于健康状态。

(5) 资源管理：

对于需要 GPU 资源的服务，你通过 `deploy` 配置来保证 Docker 可以使用它。这是在生产环境中尤其重要的，以确保应用性能。

(6) 开发工具：

利用 `x-develop` 自定义属性来支持开发过程，如自动重建和同步代码变更，使得开发和测试更加高效。

结 论

本文研究并实现了一个基于知识图谱的足球知识问答系统。通过收集和整合足球相关数据，构建了一个全面的足球知识图谱，并利用自然语言处理技术实现了对用户提出的足球问题的智能回答。系统架构设计包括数据收集与预处理、知识图谱构建、问答系统和用户界面模块，确保了系统的高效性和稳定性。

实验结果表明，该系统在准确性和响应时间上均表现良好，验证了知识图谱与大语言模型结合在智能问答中的有效性。通过 Neo4j 数据库的高效图数据存储和查询能力，系统能够处理复杂的关系数据，并通过预训练语言模型进行语义解析和答案生成，显著提升了问答系统的准确性和用户体验。

本文为知识图谱与自然语言处理在足球知识问答中的应用提供了新的思路和方法，展示了两者的结合在特定领域中的巨大潜力。未来的研究可以进一步优化系统的性能，扩展数据源，提高问答的精确度和系统的智能化水平，为用户提供更优质的服务。

尽管本文的基于知识图谱的足球知识问答系统在准确性和响应时间上表现优异，但仍有许多方面值得进一步探讨和优化。以下是几个未来的研究方向和潜在改进措施：

(1) 数据源的扩展

目前系统的数据主要来源于公开的足球数据库和网站。为了进一步提高系统的全面性和准确性，可以考虑集成更多的数据源，包括社交媒体上的实时数据、俱乐部官方发布的信息以及各类足球分析报告。此外，实时数据的引入将使系统能够提供更为及时和动态的回答。

(2) 自然语言处理技术的改进

虽然现有的自然语言处理技术已经能够较好地理解和解析用户问题，但在一些复杂或语义模糊的情况下，系统仍有提升空间。未来可以引入更多先进的自然语言处理模型，如 BERT、GPT-4 等，以提高语义理解的深度和广度。同时，通过结合上下文信息和用户历史查询记录，进一步优化问答的准确性和相关性。

(3) 知识图谱的动态更新

足球领域的信息变化迅速，比赛结果、球员状态、转会动态等都会影响知识图谱的准确性。因此，建立一个自动化的数据更新机制是非常必要的。通过定期抓取和解析最新数据，并动态更新知识图谱，系统能够保持信息的实时性和准确性。

参考文献

- [1] 方拓迁. 基于时序知识图谱的足球知识问答系统[D].广州:广州大学,2022: 1-15
- [2] 马忠贵, 倪润宇, 余开航. 知识图谱的最新进展、关键技术和挑战[J]. 工程科学学报, 2020, 42(10): 1254-1266.
- [3] Pan S, Luo L, Wang Y, et al. Unifying large language models and knowledge graphs: A roadmap[J]. IEEE Transactions on Knowledge and Data Engineering, 2024: 12-17
- [4] Baek J, Aji A F, Saffari A. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering[J]. arXiv preprint arXiv:2306.04136, 2023: 23-27
- [5] Li T, Ma X, Zhuang A, et al. Few-shot in-context learning for knowledge base question answering[J]. arXiv preprint arXiv:2305.01750, 2023: 12-15
- [6] Wang K, Duan F, Wang S, et al. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering[J]. arXiv preprint arXiv:2308.13259, 2023.13
- [7] Saxena A, Chakrabarti S, Talukdar P. Question answering over temporal knowledge graphs[J]. arXiv preprint arXiv:2106.01515, 2021: 15-17
- [8] 李源, 马新宇, 杨国利, 等. 面向知识图谱和大语言模型的因果关系推断综述[J]. 计算机科学与探索, 2023, 17(10): 2358: 12-21
- [9] Badenes-Olmedo C, Corcho O. MuHeQA: Zero-shot question answering over multiple and heterogeneous knowledge bases[J]. Semantic Web, 2023 (Preprint): 1-15.
- [10] 贾焰, 亓玉璐, 尚怀军, 等. 一种构建网络安全知识图谱的实用方法[J]. 收藏, 2018, 1:2-15
- [11] 漆桂林, 高桓, 吴天星. 知识图谱研究进展[J]. 情报工程, 2017, 3(1): 4-25.
- [12] Pan S, Luo L, Wang Y, et al. Unifying large language models and knowledge graphs: A roadmap[J]. IEEE Transactions on Knowledge and Data Engineering, 2024: 4-14
- [13] Wang C, Liu X, Song D. Language models are open knowledge graphs[J]. arXiv preprint arXiv:2010.11967, 2020: 3-16
- [14] Petroni F, Rocktäschel T, Lewis P, et al. Language models as knowledge bases?[J]. arXiv preprint arXiv:1909.01066, 2019: 4-24
- [15] Andrus B R, Nasiri Y, Cui S, et al. Enhanced story comprehension for large language models

through dynamic document-based knowledge graphs[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2022, 36(10): 10436-10444.

[16] Wang C, Liu X, Song D. Language models are open knowledge graphs[J]. arXiv preprint arXiv:2010.11967, 2020: 2-15

[17] Guo Q, Cao S, Yi Z. A medical question answering system using large language models and knowledge graphs[J]. International Journal of Intelligent Systems, 2022, 37(11): 8548-8564.

致 谢

在本论文的撰写过程中，我得到了许多人的帮助和支持，在此，我向他们表示衷心的感谢。

首先，我要感谢我的导师王晓妍副教授，她在论文的每一个阶段都给予了我无私的指导和帮助。从选题、研究方法到论文的最终修改，王老师都给予了耐心的指导和宝贵的建议，使我的研究能够顺利进行并最终完成。

其次，我要感谢我的同学和朋友们。在整个研究和写作过程中，他们与我分享了许多有价值的资源和经验，给予了我很多支持和鼓励，使我在遇到困难时能够保持积极的态度，顺利克服挑战。

特别感谢我的家人，他们一直以来对我的理解和支持，为我提供了坚实的后盾。在我专注于学术研究的过程中，他们给予了我最大的鼓励和帮助，让我能够全身心地投入到论文的写作中。

最后，我要感谢燕山大学的信息科学与工程学院，为我的研究提供了良好的学习环境和丰富的资源支持。感谢所有帮助过我的人，正是由于你们的帮助和鼓励，我才能顺利完成这篇毕业论文。

衷心感谢大家！

附录 1 开题报告

一、综述本课题国内外研究动态，说明选题的依据和意义

随着人工智能技术的迅速发展，知识图谱和自然语言处理领域特别是问答系统得到了显著的进步。知识图谱的本质是连接实体间关系的图，即揭示实体之间关系的语义网络，知识图谱作为一种组织和管理知识的有效方法，能够将世界上的事物及其相互关系以图形的方式表现出来，为问答系统提供了丰富的、结构化的知识来源。^[1]同时，近年来大语言模型，尤其是基于 Transformer architecture 的模型，如 GPT 系列和 BERT 系列，已经在自然语言理解和生成方面取得了突破性的成就。

大语言模型虽然能够生成流畅的语言回答，但在准确性和可靠性方面往往依赖于训练数据的质量和范围，LLM 通过概率模型进行推理，这是一个犹豫不决的过程。LLM 用于预测或决策的具体模式和函数对人类来说是不可直接访问或解释的。即使一些 LLM 能够通过应用思维链来解释其预测结果，它们的推理解释也会出现错觉问题^[2]，这就是知识图谱发挥作用的地方。足球知识问答系统作为一个具体应用领域，利用知识图谱处理和回答与足球相关的各种问题，包括球员信息、比赛结果、历史记录等。

1. 国内外研究动态

随着国家在体育事业上不断地投入以及体育锻炼强身健体这一观念的深入人心，人们参与体育运动的积极性也越来越高，其中尤其以足球这项运动受众最广，参与程度最高。足球运动因其观赏性，对抗性以及不可预知性，成为了世界第一运动。

然而上述常见的知识图谱项目不是专门面向足球领域，对足球领域知识只有少量涉及。因为足球数据涉及大量的时间相关信息，如比赛日期、球员转会时间等。如何在知识图谱中表示这些时间信息，就成了一个挑战。

A Saxena 等人探讨了如何使用时间知识图谱来回答自然语言问题，通过在知识图谱的每条边上提供时间范围来拓展常规知识图，方拓迁提出时序知识图谱的表示模型 STKM，用于解决常规知识图谱表示模型无法表示时序知识的问题，但为了给知识图谱中的三元组添加时间信息，使得时序知识图谱产生了大量的无实际意义但

不可缺少的三元组造成了时序知识图谱存储的有效信息的密度下降。

大语言模型虽然能够生成流畅的语言回答，但在准确性和可靠性方面往往依赖于训练数据的质量和范围，LLM 通过概率模型进行推理，这是一个犹豫不决的过程。LLM 用于预测或决策的具体模式和函数对人类来说是不可直接访问或解释的。即使一些 LLM 能够通过应用思维链来解释其预测结果，它们的推理解释也会出现错觉问题^[3]，知识图谱提供了结构化、明确和可编辑的知识表征，为缓解 LLM 的局限性提供了一种补充策略。研究人员已经探索了如何使用 KG 作为外部知识源来减轻 LLM 的幻觉。

Li 等使用 LLM 生成 SPARQL 查询的主干，并使用知识图谱填充完整的信息^[4]，Back 等人采样了包含出现问题的实体的三元组，用于 LLM 推理^[5]，Wang 等人提出了一种检索器-阅读器-验证器 QA 系统，用于访问外部知识并与 LLM 交互。

2. 选题依据及意义

技术成熟度：知识图谱和问答系统的相关技术已日趋成熟^[6]，为足球知识问答系统提供了技术基础。数据丰富度：足球领域拥有大量的历史数据、统计信息和实时数据，适合构建知识图谱。应用需求：足球迷和相关工作者对快速获取准确足球信息的需求日增，推动了足球知识问答系统的开发提高信息获取效率：足球知识问答系统能够快速回答用户关于足球的各种查询，大大提高信息获取的效率。促进足球知识普及和教育：系统可以作为足球知识的普及工具，帮助人们学习和理解足球历史、规则和文化。增强体验：对于足球迷而言，能够即时获取关于喜爱球队或球员的深度信息，增强了观赛体验和参与感。

二、研究的基本内容，拟解决的主要问题

1. 研究的基本内容

本研究将实现一个基于基于知识图谱的足球知识问答系统，利用知识图谱与大语言模型相结合，利用两者技术优势建立出解决知识图谱时间处理困难的问题和大语言模型出现幻读的问题^[7]，同时与 Web 技术相结合，实现处理和回答与足球相关的各种问题，包括球员信息、比赛结果、历史记录等。

2. 拟解决的主要问题

该研究拟解决的主要问题包括：

（1）知识图谱时间处理困难

足球知识问答系统需要处理大量与时间相关的数据，如比赛日期、球员转会时间等。知识图谱在处理这类时间敏感信息时面临着挑战，如时间信息的表示、存储和查询等。研究将探索有效的时间信息处理方法，以提高知识图谱在足球领域的应用效率和准确性。

(2) 传统的足球问答系统可能要求用户了解特定的查询语法或者使用精确的关键词来提出问题，这对于非技术用户来说可能既困难又不直观。足球数据非常丰富和多样，涵盖比赛结果、球员统计、转会信息等多个维度，传统查询方式难以有效处理这些复杂的数据关系和多维度的信息需求。

(3) 大语言模型的幻读问题：在利用大语言模型进行问答时，可能会遇到基于错误或过时信息生成答案的问题，这种现象被称为幻读。研究将探讨如何减少幻读现象，通过结合知识图谱提供的结构化、可靠信息，提高问答系统的准确性和可靠性。

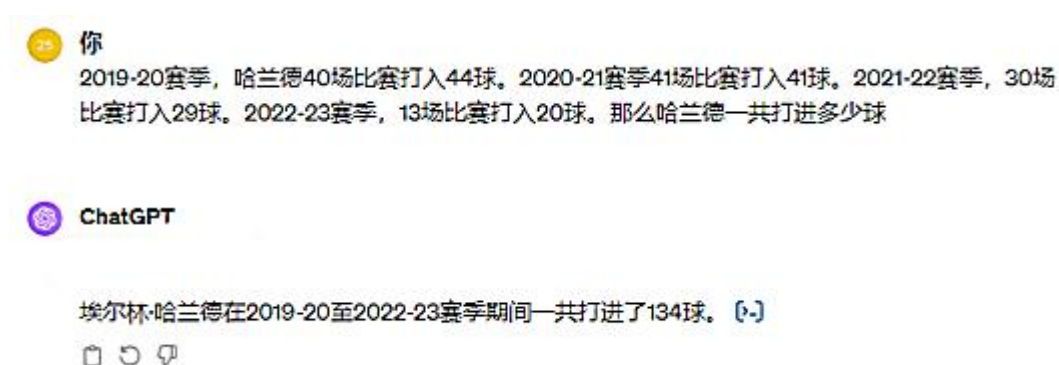


图 1 使用来自知识图谱的信息

(4) 知识图谱与大语言模型的有效结合：如何有效地结合知识图谱和大语言模型的技术优势，是本研究的核心问题之一。研究将探索两者之间的交互机制，如利用知识图谱增强大语言模型的语义理解能力，同时利用大语言模型弥补知识图谱在自然语言处理方面的不足。

(5) 与 Web 技术的整合：为了实现足球知识问答系统的广泛可访问性和用户友好性，研究还将探讨如何将知识图谱和大语言模型与 Web 技术相结合，包括前端界面设计、后端数据处理、以及用户交互方式等，以提供一个易于使用、响应迅速的在线问答平台。

二、研究步骤、方法及措施

1. 研究步骤

使用LLMs应用查询知识图谱

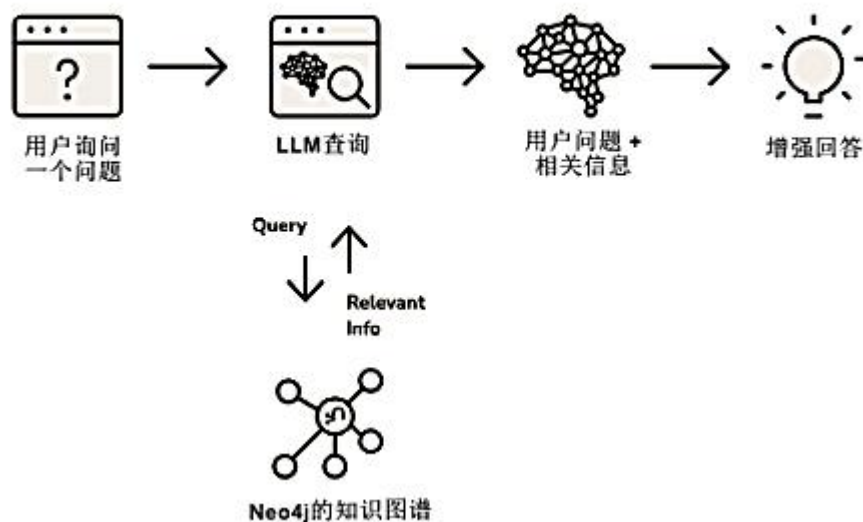


图 2 使用 LLMs 应用查询知识图谱

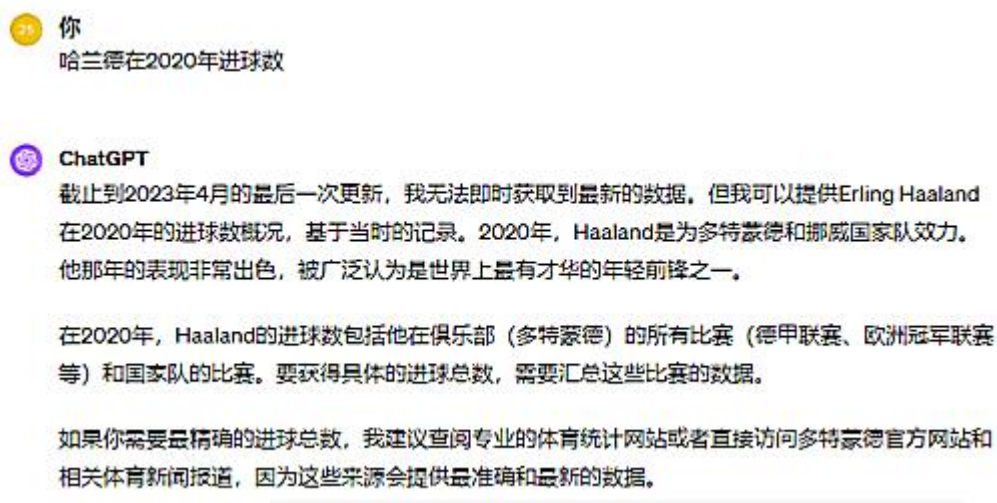


图 3 未使用来自知识图谱的信息

（1）知识图谱构建：首先需要构建或使用现有的涵盖足球相关信息的知识图谱。这可能包括实体如球员、球队、联赛、比赛、比分等。同时也需要定义这些实体之间的关系，例如“效力于”、“是…的成员”、“参加了”等。

（2）数据来源：从多种数据源中填充知识图谱，包括体育数据库、足球协会的官方网站、新闻报道、比赛统计和历史记录等。

(3) 数据处理：通过知识抽取技术，可从已有的结构化、半结构化、非结构化样本源以及一些开源的百科类网站抽取实体、关系、属性等知识要素。通过知识融合，可消除实体、关系、属性等指称项与实体对象之间的歧义，得到一系列基本的事实表达。通过本体抽取、知识推理和质量评估形成最终的知识图谱库。按照知识抽取、知识融合、知识推理 3 个步骤对知识图谱迭代更新，实现碎片化的互联网知识的自动抽取、自动关联和融合、自动加工，从而拥有词条自动化链接、词条编辑辅助功能，最终达成全流程自动化知识获取的目标[8]。

(4) 图谱查询：根据用户的问题，系统需要能够构建和执行针对知识图谱的查询。

(5) 大语言模型处理：根据查询出的结果，使用大模型以辅助处理，呈现出完整清晰的回答。

(6) 用户界面：最后，为了让用户能够方便地提问和查看答案，需要开发一个友好的用户界面。这可以是一个网页应用、移动应用或者与聊天机器人集成的界面。

2. 技术栈选择

(1) 知识图谱的表示

OWL (Web 本体语言)

OWL 是一种用于定义和实例化 Web 本体的语言，它基于 RDF，并提供了更丰富的数据描述能力，使得能够表示复杂的关系如同源性、分类、属性、继承等。OWL 使得数据不仅可以被描述和关联，还可以通过逻辑推理引擎进行复杂的查询和推理，增强了知识的表达能力。

OWL 定义了三种子语言，适用于不同复杂度的本体建模需求：

OWL Lite：支持简单的分类层次以及简单的约束特性。

OWL DL (描述逻辑)：提供了完整的 OWL 语言特性，保证了计算的可判定性和完备性。

OWL Full：提供了最大的表达能力，允许本体中的类和属性作为个体进行处理，但不保证计算的可判定性。

使用 OWL，可以定义类（概念）、属性（关系）和个体（实例），并通过逻辑推理发现数据间的隐含关系。例如，可以定义一个类“人”，一个子类“学生”，并声明某个个体“小明”属于“学生”这个类。通过逻辑推理，我们可以得出“小

明”也属于“人”这个更广泛的类。

RDF 和 OWL 共同构成了构建语义网和知识图谱的基础，它们使得知识的表达、共享和推理成为可能，为数据的互联网化和智能化应用提供了强大的支持。

（2）图数据库

Neo4j 是一个高性能的图数据库管理系统，专为处理复杂数据关系而设计。它是一个基于图的数据库，这意味着数据模型主要是围绕节点(Nodes)、关系(Relationships)和属性(Properties)构建的。Neo4j 以其优秀的连接数据处理能力而著称，特别适合于需要深入分析数据关系的场景，如社交网络、推荐系统、欺诈检测、知识图谱



图 4 Neo4j

特点和优势：

高性能和灵活性：Neo4j 优化了处理连接数据的查询性能，尤其是在执行深度连接查询时，相比传统的关系数据库有显著的性能优势。

Cypher 查询语言：Neo4j 使用自己的查询语言 Cypher，它是一种声明式的图形查询语言，使得查询和更新图形数据变得简单直观。

事务支持：支持 ACID 事务（原子性、一致性、隔离性、持久性），确保数据的安全性和一致性。

可扩展性：虽然 Neo4j 是一个原生图数据库，专为处理高度连接的数据设计，但它也提供了一定的可扩展性，支持大规模数据的处理。

社区和生态系统：Neo4j 拥有一个活跃的开发社区和丰富的生态系统，包括各种工具、库和框架，以及丰富的学习资源。

核心概念

节点 (Nodes)：数据的基本元素，可以包含多个属性。节点通常用来表示实体，如人、地点、事物等。

关系 (Relationships)：连接节点的线，表示节点之间的关系。每个关系都有一个类型，并且可以包含属性。

属性 (Properties)：节点和关系可以包含属性，这些属性以键值对的形式存储数据。

标签 (Labels)：可以给节点添加一个或多个标签，以便对不同的节点进行分类。

应用场景

Neo4j 被广泛应用于各种场景，包括但不限于：

社交网络分析：分析用户之间的关系，如朋友、关注等。

推荐系统：基于用户的行为和偏好，推荐相关的产品或内容。

欺诈检测：通过分析交易和用户之间的关系，识别潜在的欺诈行为。

知识图谱：构建和查询复杂的知识库，支持语义搜索和信息检索。

网络和 IT 运维：分析网络和系统中的依赖关系，优化性能和故障排除。

(3) web 框架

Vue+Node.js 框架

Vue 的优点：作为一款轻量级框架，简单易学，容易上手，还便于与第三方库进行整合；可以对数据进行双向绑定；可以进行组件化开发，提高开发效率。

Node.js：Node.js 使得 JavaScript 既可以在浏览器的基础上运行，也可以脱离浏览器运行。语法完全是 js 语法，开发成本和学习成本较低，开发速度快；提供了异步功能。其中 Express 是最受欢迎的 Node.js 框架，基于 MVC 架构。

Vue+SpringBoot+MyBatis/MyBatis-Plus

SpringBoot 的优点：SpringBoot 提供了自动配置 Spring、内嵌 Servlet 容器、对运行项目进行监控等功能，它继承了 Spring 的良好优点，简化了依赖与配置，使开发者可以尽快地启动 Spring 应用程序，提高了部署与开发的效率。

MyBatis 的优点：MyBatis 是一款持久化框架。相比于 JDBC，代码量较小；简单易学，不会对数据库现有设计强加任何影响；MyBatis 提供了 XML 标签，支持编写动态 SQL 语句；SQL 写在了 XML 中，可以降低 SQL 语句与程序代码的耦合度，

提高了可重用性。

MyBatis-Plus: 在 MyBatis 的基础上做了增强，提高了开发效率。

(4) Langchain

LangChain 提供了对几个主要模块的支持。

模型（models）：LangChain 支持的各种模型类型和模型集成。

提示（prompts）：包括提示管理、提示优化和提示序列化。

内存（memory）：内存是在链/代理调用之间保持状态的概念。LangChain 提供了一个标准的内存接口、一组内存实现及使用内存的链/代理示例。

索引（indexes）：与您自己的文本数据结合使用时，语言模型往往更加强大——此模块涵盖了执行此操作的最佳实践。

链(chains)：链不仅仅是单个 LLM 调用，还包括一系列调用（无论是调用 LLM 还是不同的实用工具）。LangChain 提供了一种标准的链接口、许多与其他工具的集成。LangChain 提供了用于常见应用程序的端到端的链调用。

代理（agents）：代理涉及 LLM 做出行动决策、执行该行动、查看一个观察结果，并重复该过程直到完成。LangChain 提供了一个标准的代理接口，一系列可供选择的代理，以及端到端代理的示例。

(5) GPTS

OpenGPTs 的简介

2023 年 11 月 6 日，LangChain 官方发布了 OpenGPTs 项目，这是一个令人兴奋的全新开源项目，允许您创建定制的聊天机器人，比 OpenAI 等封闭解决方案更具灵活性。基于 LangChain、LangServe 和 LangSmith 构建，OpenGPTs 使您能够控制用于驱动对话人工智能的语言模型、工具和 API。通过 OpenGPTs，您可以配置和构建根据您的需求定制的聊天机器人。

与直接使用 OpenAI 相比，OpenGPTs 的吸引力在于它更具可定制性。具体而言，您可以选择使用哪些语言模型，以及更轻松地添加自定义工具。您还可以直接使用底层 API 并选择构建自己的自定义 UI。

OpenGPTs 为创建根据您的需求定制的聊天机器人提供了一个开放的框架。透明的代码库和 LangChain 集成为您提供了无与伦比的控制和可定制性。这是一个创建与 OpenAI 的 GPT 和 Assistants API 类似体验的开源工作。它基于 LangChain、

LangServe 和 LangSmith。

主要特点

OpenGPTs 旨在提供与 OpenAI 相当的功能，同时支持定制：

Sandbox 沙盒：导入、测试和修改以代码定义的现有聊天机器人

Custom Actions 自定义操作：通过 OpenAPI 规范扩展聊天机器人的功能

Tools 工具：整合自定义工具，用于网页浏览、图像生成等

Analytics 分析：通过 LangSmith 集成监控使用数据

Drafts & Sharing 草稿和分享：保存、分享和部署已完成的聊天机器人

Marketplace 市场：分发并从社区中找到聊天机器人

2. 研究实现

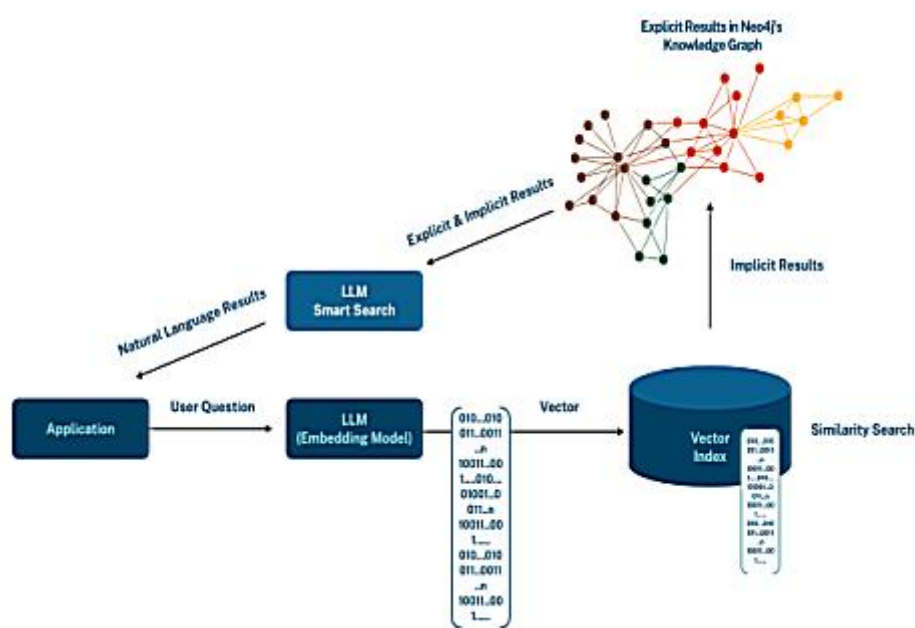


图 5 langchain 的向量化和知识图谱结合

- (1) 在 LangChain、LlamaIndex 等编排框架中使用 Neo4j
- (2) 为 Neo4j 知识图中的数据添加向量嵌入并为其建立索引
- (3) 为用户输入生成嵌入
- (4) 在向量索引中通过相似性搜索找到最相关的节点并从知识图中检索上下文信息

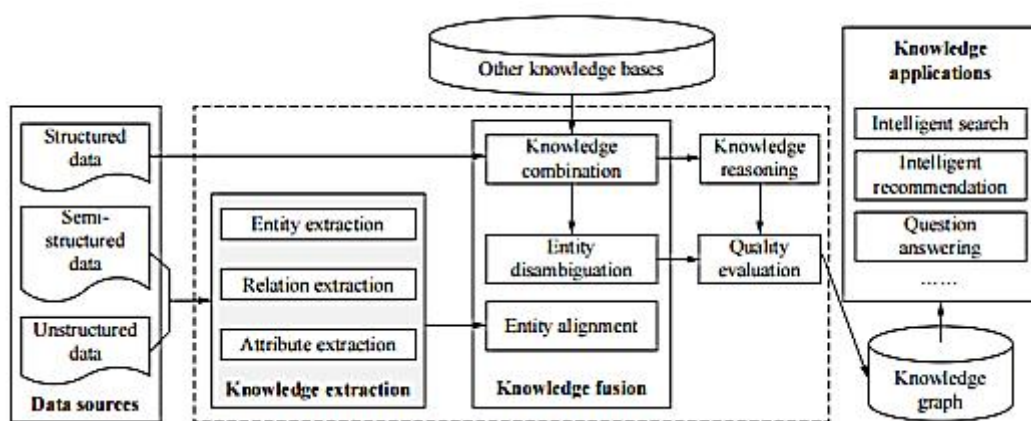


图 6 知识图谱相关技术

(5) 使用 llm 为用户问题以进行自然语言搜索

(6) 围绕着 llm 对查询出来的上下文信息做信息增强

四、研究工作进度

1-3 周：完成需求分析，广泛收集和阅读相关材料，了解选题的背景和意义，学习选题涉及到的部分技术及理论知识，为后续的设计与开发做好铺垫。进行开发环境的搭建，完成开题报告和开题答辩的工作。

4-9 周：继续学习技术与理论知识，完成系统总体设计，进行系统详细设计，设计数据库。对系统进行编码实现，完成系统基础功能的开发，完成相应的测试工作，编写论文初稿并完成中期检查。

10-13 周：针对检查结果，继续进行系统的开发、测试与完善，保证系统的实用性、稳定性与安全性。结合相关文档、开发过程及中期检查结果等内容完善毕业论文，完善论文初稿。

14-15 周：对毕业论文进行修改完善，完成毕业论文定稿，制作答辩用 PPT，准备最终答辩。

五、主要参考文献

- [1] 方拓迁. 基于时序知识图谱的足球知识问答系统[D].广州:广州大学,2022: 1-15
- [2] 马忠贵, 倪润宇, 余开航. 知识图谱的最新进展、关键技术和挑战[J]. 工程科学学报, 2020, 42(10): 1254-1266.
- [3] Pan S, Luo L, Wang Y, et al. Unifying large language models and knowledge graphs: A roadmap[J]. IEEE Transactions on Knowledge and Data Engineering, 2024: 12-17

- [4] Baek J, Aji A F, Saffari A. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering[J]. arXiv preprint arXiv:2306.04136, 2023: 23-27
- [5] Li T, Ma X, Zhuang A, et al. Few-shot in-context learning for knowledge base question answering[J]. arXiv preprint arXiv:2305.01750, 2023: 12-15
- [6] Wang K, Duan F, Wang S, et al. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering[J]. arXiv preprint arXiv:2308.13259, 2023.13
- [7] Saxena A, Chakrabarti S, Talukdar P. Question answering over temporal knowledge graphs[J]. arXiv preprint arXiv:2106.01515, 2021: 15-17
- [8] 李源, 马新宇, 杨国利, 等. 面向知识图谱和大语言模型的因果关系推断综述[J]. 计算机科学与探索, 2023, 17(10): 2358: 12-21
- [9] Badenes-Olmedo C, Corcho O. MuHeQA: Zero-shot question answering over multiple and heterogeneous knowledge bases[J]. Semantic Web, 2023 (Preprint): 1-15.
- [10] 贾焰, 亓玉璐, 尚怀军, 等. 一种构建网络安全知识图谱的实用方法[J]. 收藏, 2018, 1:2-15

附录 2 中期报告

一、任务书中本阶段工作目标与任务要求

基于知识图谱的足球知识问答系统，是一组由 Docker Compose 协调的 Docker 容器组成，其中包括本地 LLMs 的管理工具（Ollama）、用于接地的数据库（Neo4j）和基于 LangChain 的应用程序。这些容器提供了一个支持代理应用（RAG 应用程序背后的理念是在查询时为 LLMs 提供额外的上下文，以便回答用户的问题）程序的开发环境，其中包含数据导入和响应生成用例。尝试在知识图谱中导入相关的足球信息，并检查底层基础信息的多样性如何影响用户界面中 LLM 生成的响应。基于知识图谱的足球知识问答系统包括：

1. 应用程序容器（Python 应用程序逻辑，使用 LangChain 进行协调，使用 Streamlit 进行用户界面）。
2. 具有向量索引和图搜索功能的数据库容器（Neo4j）。
3. LLM 容器 Ollama。

本阶段工作工作目标与任务要求如下：

根据之前确认的执行计划，进行系统的详细设计工作，确保所有功能和要求在设计阶段得到详尽的体现。

遵循系统设计文档的指导，进行软件的编码工作。

持续推动项目按计划前进，及时识别和总结项目执行中遇到的问题，并制定相应的解决策略。

严格管理项目的版本控制，系统地整理和保存项目中产生的各种文档资料，为撰写最终的学术论文做好充分准备。

二、目前已完成任务情况

1. 总体项目进度

目前为止基于知识图谱的足球知识问答系统，进行系统详细设计如图 1 所示，设计了以下服务：

- (1) bot 拥有经典的 LLM 聊天用户界面，用户可以提出问题并获得答案。
- (2) llm_chatbot 利用 GPT 模型将纯英文文本查询自动转换为 Cypher(知识图谱的查询语句)。

- (3) 知识图谱数据库
- (4) 前后端分离的基于知识图谱的足球知识问答系统。

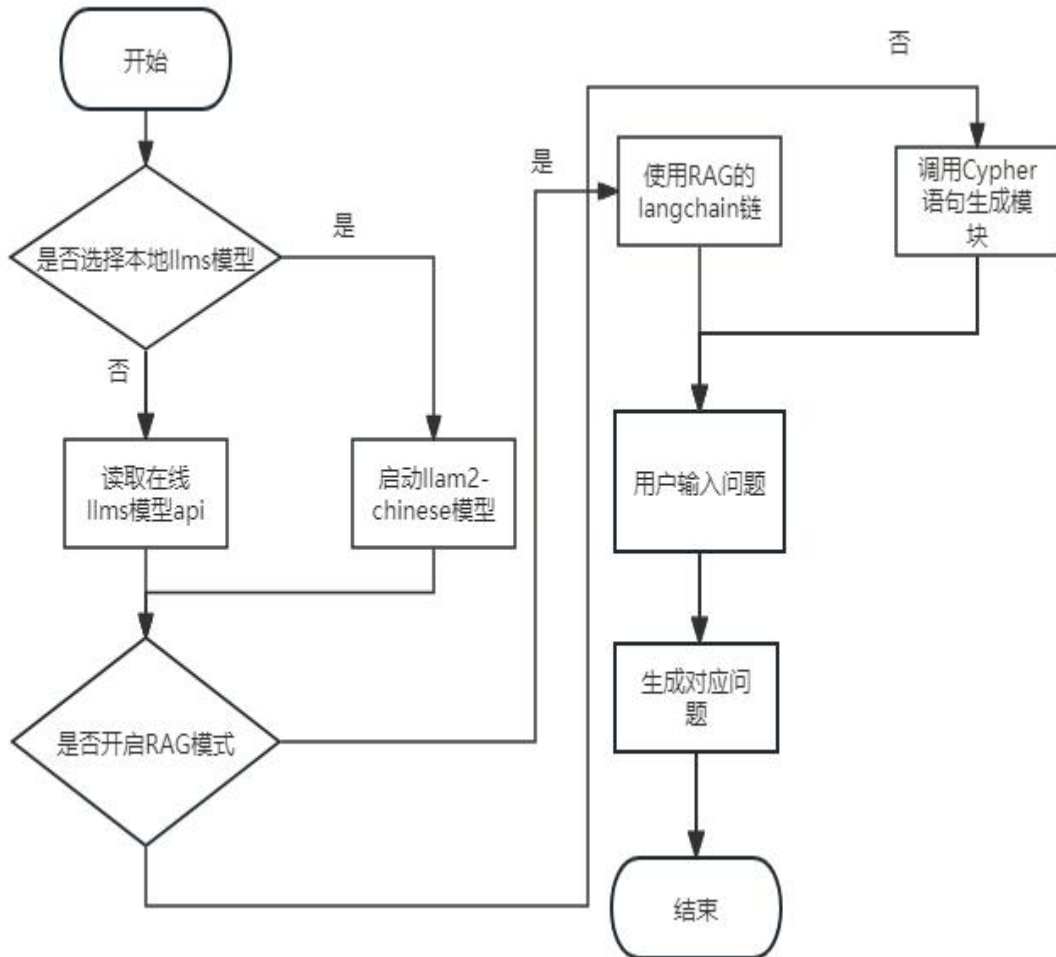


图 1 主要系统流程图

2. 详细设计原理与成果介绍

(1) Neo4j 数据库设计

通过爬虫等手段爬取英超的数据，导入到 Neo4j 数据库，拥有 15485 条关系，10 个主要节点的图数据库,下面分别介绍主要的图结构。

① 球员信息（5000 条数据）

如图 2 所示，主要节点有 3 个，分别为球员（player），所属国家（country），所在俱乐部（club），关系包括球员出生于（born_in）某个国家和球员效力(plays_for)于某个球队

② 比赛信息 （2001 年到 2023 的英超数据）

如图 3 所示，主要节点包括赛区（Division），球队（Team ），比赛（Match），裁判(Referee)

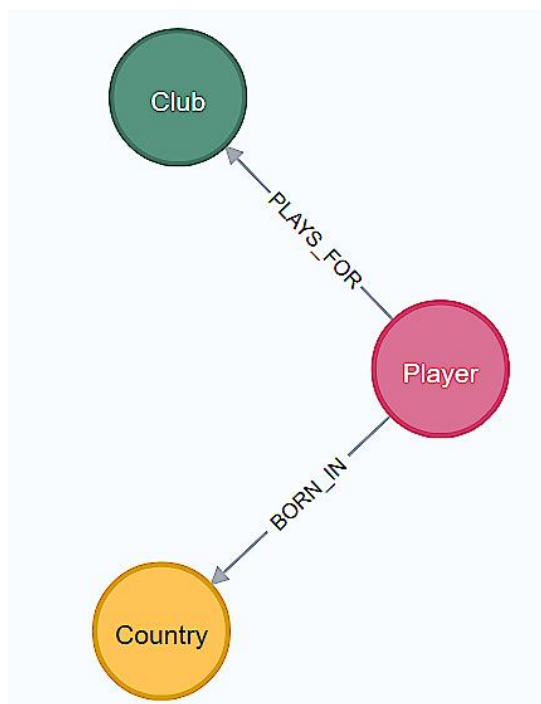


图 2 neo4j 部分图结构

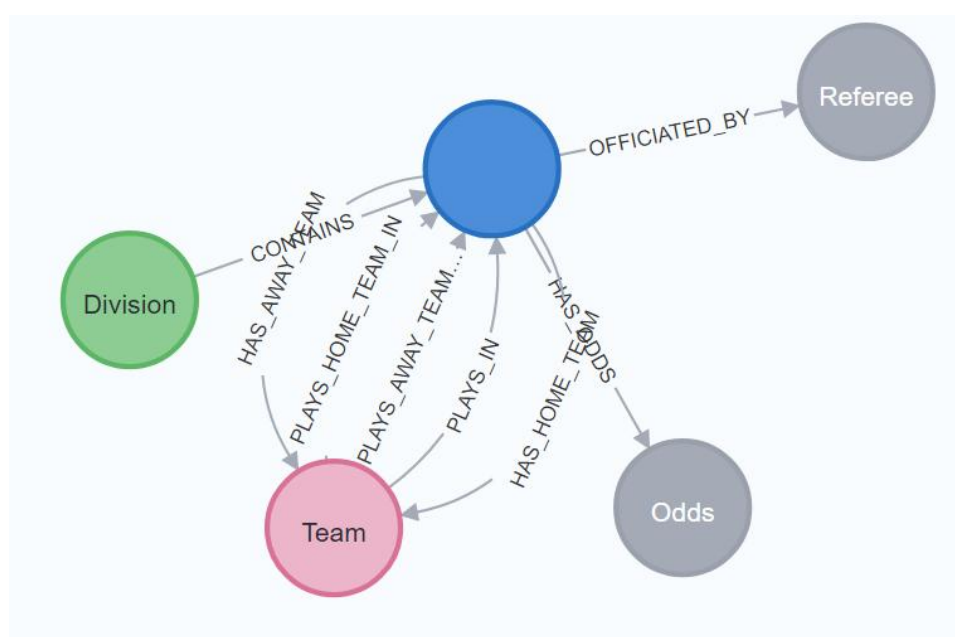


图 3 比赛信息

③ 某个国家所有的所有球员

如图 4 某 country 的部分关系图,使用 MATCH (n:Country) RETURN n LIMIT 25

④ 下面是数据库部分设计

节点类型（Entities）

Division (赛区) , Match (比赛) , Team (球队) , Referee (裁判)

关系类型（Relationships）

Division-CONTAINS->Match (赛区 包含 比赛) ,
Match-HAS_HOME_TEAM->Team (比赛有主队) , Match-HAS_AWAY_TEAM->Team
(比赛有客队) , Match-OFFICIATED_BY->Referee (比赛由裁判主持) ,
Team-PLAYS_IN->Division (球队在某赛区比赛)

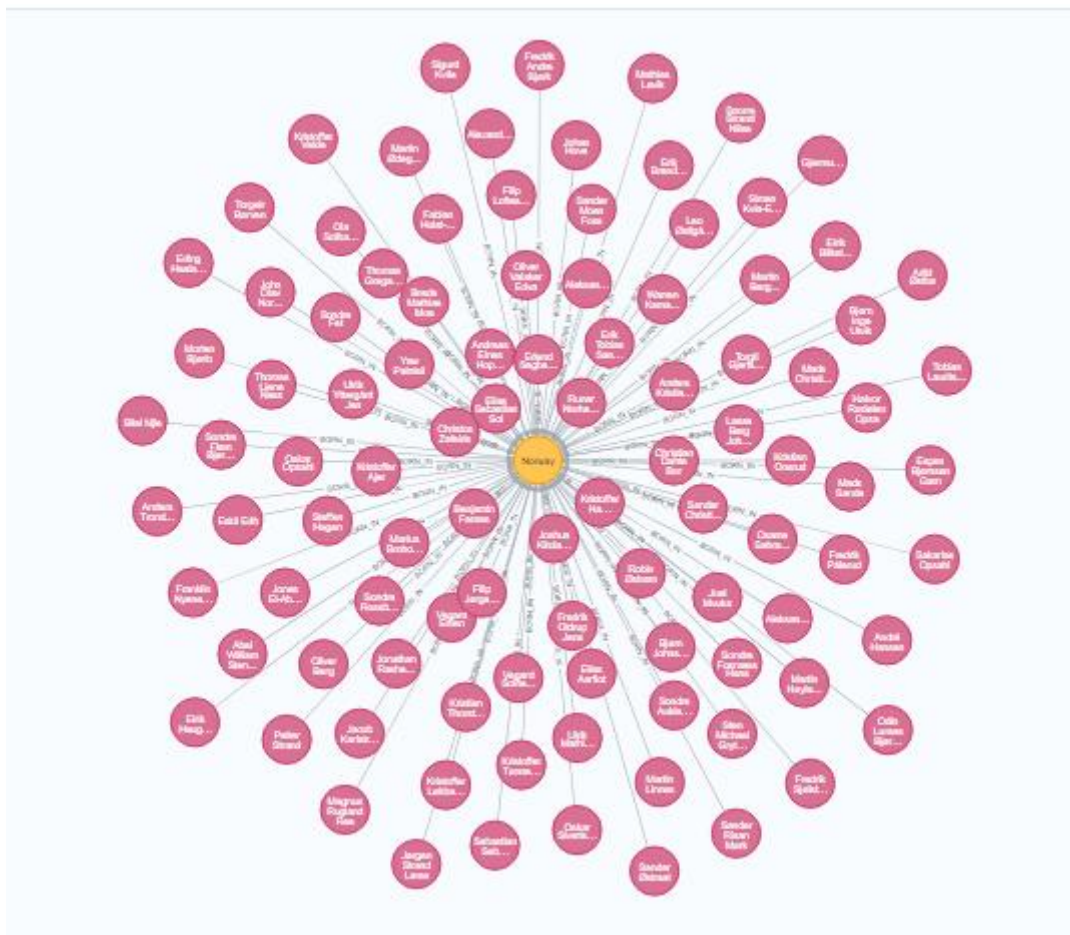


图 4 某 country 的部分关系图

具体属性（Attributes）

Match , Date (日期) , Full Time Home Goals (全场主队进球) , Full Time Away

Goals (全场客队进球) , Half Time Home Goals (半场主队进球) , Half Time Away Goals (半场客队进球) , Attendance (观众人数) , Match statistics such as shots, fouls, cards (比赛统计: 射门、犯规、黄牌等)

```
LOAD CSV WITH HEADERS FROM 'file:///datas/data1/CLEAN_FIFA23_official_data.txt' AS line FIELDTERMINATOR ','
MERGE (player:Player {
  id: line.ID,
  name: line.Name,
  age: toInteger(line.Age),
  datasetYear: 2023, // 这里直接设置为数据集的年份
  photo: line.Photo,
  nationality: line.Nationality,
  value: toFloat(line.ValueGBP),
  wage: toFloat(line.WageGBP),
  preferredFoot: line.PreferredFoot,
  position: line.Position,
  joined: line.Joined,
  contractValidUntil: toFloat(line.ContractValidUntil),
  height: toFloat(line.Height),
  weight: toFloat(line.Weight),
  yearJoined: toInteger(line.Year_Joined)
})
MERGE (club:Club {name: line.Club})
ON CREATE SET club.Logo = line.ClubLogo
MERGE (player)-[:PLAYS_FOR {joined: line.Joined}]->(club)
MERGE (country:Country {name: line.Nationality})
ON CREATE SET country.flag = line.Flag
MERGE (player)-[:COMES_FROM]->(country)
```

图 5 部分 Cypher 语句

3. 本地大语言模型

近来, 开源 LLM 研究取得了显著进展。Llama2 和 Mistral 等模型显示出令人印象深刻的准确性和性能水平, 使它们成为商业模型的可行替代品。使用开放源码 LLMs 的一个显著好处是消除了对外部 LLM 提供商的依赖, 同时保留了对数据流以及数据共享和存储方式的完全控制。Ollama 项目的维护者认识到了开源 LLMs 的机遇, 他们提供了一种无缝解决方案, 可在您自己的基础架构甚至笔记本电脑上设置和运行本地 LLMs。

本项目使用了 llama2-chinese 模型, 这个模型是基于 Meta Platform, Inc. 所发布的 Llama 2 Chat 开源模型来进行微调。根据 Meta, Llama 2 的训练数据达到了两万亿个 token, 上下文长度也提升到 4096。对话上也是使用 100 万人类标记的数据微调。

由于 Llama 2 本身的中文对齐比较弱, 开发者采用了中文指令集来进行微调, 使其具备较强的中文对话能力。目前这个中文微调参数模型总共发布了 7B, 13B 两种参数大小。

| 排名 | 模型 | 胜和率 | 语义理解与抽取 | 闲聊 | 上下文对话 | 角色扮演 | 知识与百科 | 生成与创作 | 代码 | 逻辑与推理 |
|----|---------------------------|-------|---------|-------|-------|--------|-------|--------|-------|--------|
| - | gpt-4 | 94.64 | 80.00 | 97.30 | 93.18 | 100.00 | 87.76 | 100.00 | 97.92 | 100.00 |
| - | Baichuan-13B-Chat | 65.28 | 45.00 | 88.33 | 78.33 | 91.67 | 55.00 | 91.67 | 25.00 | 50.88 |
| - | ChatGLM2-6B | 36.50 | 33.33 | 38.33 | 36.67 | 41.67 | 20.00 | 40.00 | 21.67 | 55.00 |
| 1 | openbuddy_llama2_13b | 35.12 | 33.33 | 40.00 | 23.33 | 20.00 | 23.33 | 46.67 | 33.33 | 58.62 |
| 2 | Llama-2-13B-chat | 27.05 | 43.33 | 35.00 | 27.12 | 11.67 | 15.00 | 46.67 | 6.67 | 35.00 |
| 3 | FlagAlpha-Llama2-13b-Chat | 26.51 | 33.33 | 36.67 | 36.67 | 24.14 | 10.00 | 50.00 | 6.67 | 41.38 |
| 4 | Chinese-Alpaca-2-7B | 14.43 | 20.00 | 6.67 | 20.00 | 10.34 | 6.67 | 3.33 | 10.00 | 37.93 |
| 5 | firefly_llama2_13b | 12.54 | 16.67 | 6.90 | 3.57 | 3.33 | 0.00 | 6.67 | 16.67 | 46.67 |
| 6 | FlagAlpha-Llama2-7b-Chat | 12.50 | 17.24 | 20.69 | 16.67 | 3.33 | 6.67 | 13.33 | 3.33 | 26.67 |
| 7 | yayi_13b_llama2 | 8.78 | 16.67 | 0.00 | 10.00 | 3.33 | 3.45 | 3.33 | 10.34 | 20.00 |

图 16 2023 年 8 月 SuperCLUE 开放式多轮测评-Llama2 中文开源模型排行榜

如图 6 所示，在 SuperCLUE 开放式多轮测评-十大能力成绩评估中，我们发现 Llama2 中文模型（FlagAlpha-Llama2-7b-Chat）在多数任务上效果总体效果还比较一般，多数能力的平均分及及格线都有比较大的差距。

但胜在拥有比较不错的中文回答，如图 7 所示，和 7B 大小的参数,实际效果还可以。



图 7 问答系统（翻译为：查尔顿、切尔西、考文垂、德比郡、利兹联、莱斯特城、利物浦、桑德兰、托特纳姆热刺和曼联是英超球队）

并且本项目通过构建一个 dockerfile 文件来拉取本地大语言模型，实现了本地大

语言模型的拉取的封装。



图 8 使用 llama-chinese 模型

4. Docker-compose 的构建

本项目利用了 Docker 和 Docker Compose 来构建和管理一个由多个服务组成的复杂应用。这种方法允许单独配置和管理各个组件，实现服务的解耦合和灵活部署；如图 9 所示。

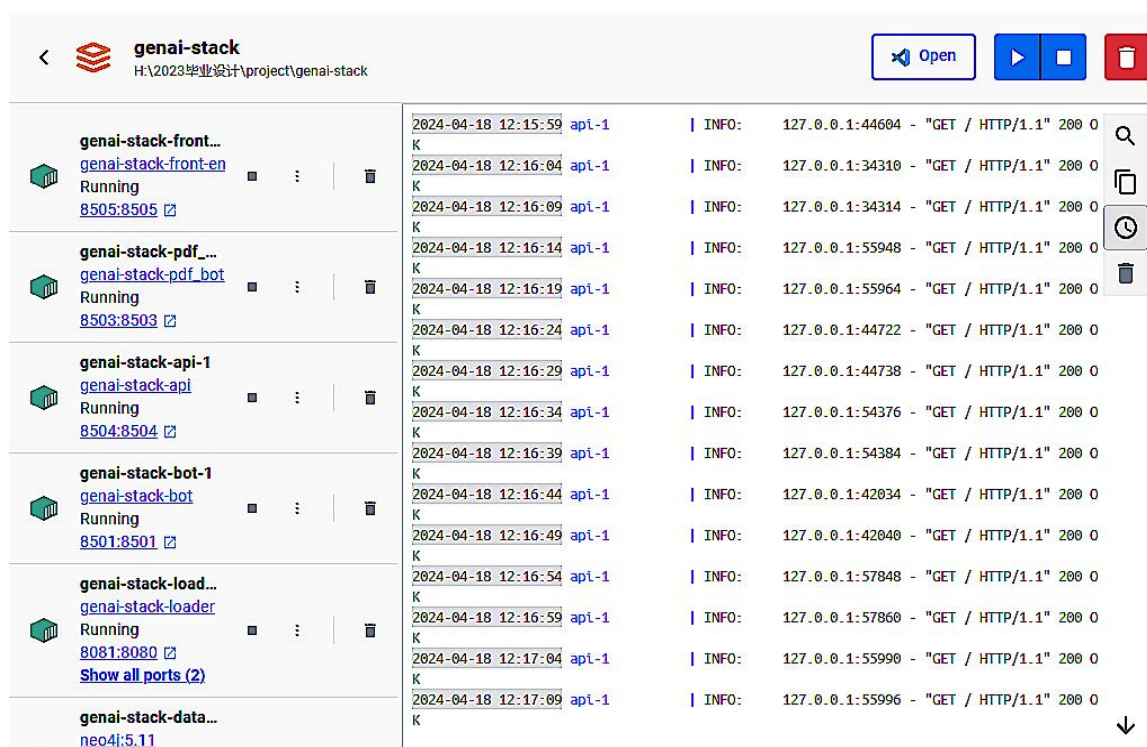


图 9 docker desktop 运行图

(1) 服务定义：

定义了多个服务，每个服务都可以单独构建和配置。这包括基本的的服务（如 API 服务、前端服务）、数据库服务（如 Neo4j）、以及特定的功能服务（如模型

拉取服务）。

（2）环境配置：

通过环境变量，为每个服务提供了必要的配置，这些变量可以在运行时进行调整。这包括数据库连接信息、API 密钥、服务 URL 等。

（3）网络配置：

所有服务都连接到同一个 Docker 内部网络 (net)，这样它们就可以互相发现和通信。

（4）依赖管理：

使用 `depends_on` 属性来管理服务间的依赖关系，确保例如数据库和模型拉取服务在其他依赖它们的服务之前启动并且处于健康状态。

（5）资源管理：

对于需要 GPU 资源的服务，你通过 `deploy` 配置来保证 Docker 可以使用它们。这是在生产环境中尤其重要的，以确保应用性能。

（6）开发工具：

利用 `x-develop` 自定义属性来支持开发过程，如自动重建和同步代码变更，使得开发和测试更加高效。

5. Bot 项目

此外，Neo4j 几天前才添加了向量索引搜索，这使其更接近支持基于非结构化文本的 RAG 应用。

1. 使用 LangChain 文档阅读器阅读维基百科文章
2. 将文本分块
3. 将文本存储到 Neo4j 中并使用新添加的向量索引对其进行索引
4. 实施问答工作流程以支持 RAG 应用程序。

拥有经典的 LLM 聊天用户界面，如图 19 所示，用户可以提出问题并获得答案。这里有一个名为 RAG 模式的开关，用户可以完全依赖 LLMs 训练有素的知识（RAG：已禁用），也可以使用功能更强的模式（RAG：已启用），即应用程序使用文本嵌入和图形查询进行相似性搜索，在数据库中找到最相关的问题和答案。

```

query = "足球俱乐部是什么的"

results = neo4j_vector.similarity_search(query, k=5)
print(results[0].page_content)

```

足球俱乐部足球职业化及专业化的一个标志，是足球队以足球谋生时所被聘用的机构。

== 历史 ==

第一家现代足球俱乐部是英国谢菲尔德足球俱乐部。全亚洲最历史悠久的又运作至今的足球

== 经营 ==

足球俱乐部的收入主要依靠出售门票、经营广告、转让转播权、足球彩票、公司赞助、办各

== 参见 ==

足球俱乐部列表

图 10 通过相似性搜索 RAG 查询问题

6. 使用 LLMs 为 neo4j 提供自然语言接口

```

def configure_llm_only_chain(llm):
    # LLM only response
    template = """
    You are a helpful assistant that helps a support agent with answering programming questions.
    If you don't know the answer, just say that you don't know, you must not make up an answer.
    """
    system_message_prompt = SystemMessagePromptTemplate.from_template(template)
    human_template = "{question}"
    human_message_prompt = HumanMessagePromptTemplate.from_template(human_template)
    chat_prompt = ChatPromptTemplate.from_messages(
        [system_message_prompt, human_message_prompt]
    )

    def generate_llm_output(
        user_input: str, callbacks: List[Any], prompt=chat_prompt
    ) -> str:
        chain = prompt | llm
        answer = chain.invoke(
            {"question": user_input}, config={"callbacks": callbacks}
        ).content
        return {"answer": answer}

    return generate_llm_output

```

图 11 基于大语言模型的查询

Neo4j 一直并且非常擅长在 RAG 应用程序中存储和分析结构化信息。

可以直接向 LLM 提供 schema 信息，让它仅基于图 schema 信息构建 Cypher 语句，run 函数首先生成一个 Cypher 语句。然后，使用生成的 Cypher 语句来查询

Neo4j 数据库。如果 Cypher 语法有效，则返回查询结果。但是，假设存在 Cypher 语法错误。在这种情况下，我们对 GPT-4 进行一次后续操作，提供在先前调用中构造的生成的 Cypher 语句，并包含来自 Neo4j 数据库的错误。GPT-4 非常擅长修复出现错误的 Cypher 语句。

（1）使用 langchain 连接 noe4j 数据库，连接到 Neo4j 实例，并在初始化时获取模式信息。然后，图模式信息可用作 GPT-4 模型的输入。

图形模式以字符串格式存储，结构如下：

```
f"This is the schema representation of the Neo4j database.  
Node properties are the following:  
{node_props}  
Relationship properties are the following:  
{rel_props}  
Relationship point from source to target nodes  
{rels}  
Make sure to respect relationship types and directions"
```

（2）接下来做一些提示工程，为 GPT-4 模型创建一个系统提示，定义一个生成 Cypher 语句的函数

```
def construct_cypher(self, question, history=None):  
    messages = [  
        {"role": "system", "content": self.get_system_message()},  
        {"role": "user", "content": question},  
    ]  
    # Used for Cypher healing flows  
    if history:  
        messages.extend(history)  
  
    completions = openai.ChatCompletion.create(  
        model="gpt-4",  
        temperature=0.0,  
        max_tokens=1000,  
        messages=messages  
    )  
    return completions.choices[0].message.content
```

图 12 生成 Cypher 语句

(3) 生成 Cypher 语句，如图 13 所示

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
cypher
MATCH (u:User {display_name: "A. L"})-[:ASKED]->(q:Question)
RETURN q ORDER BY q.view_count DESC LIMIT 1
```

图 13 使用 LLMs 生成 Cypher 语句

7. 前后端项目

(1) 本项目旨在开发一个具有与现有应用 bot 相同功能的新应用程序。不同于原有的架构，新的应用程序将采用当前最先进的前端和后端分离技术，以提升性能、可维护性和用户体验。通过采用这种现代化的方法，项目团队预期能够提供一个更灵活、易于扩展的解决方案，同时降低未来维护的复杂性和成本。

(2) 前端技术：使用 Vite、Svelte 和 Tailwind CSS。

(3) Vite：作为现代前端开发环境和构建工具，Vite 提供了极快的冷启动、即时热模块更新和真正的按需编译，从而极大地提高开发效率。

(4) Svelte：一个新兴的前端框架，以其简洁的语法和编译时的优化著称，允许写更少的代码同时运行更快，提供了一种更直接的方式来构建动态界面。

(5) Tailwind CSS：一个实用主义的 CSS 框架，用于快速定制设计，它允许开发者通过功能类直接在标记中构建设计，从而加快开发速度并减少样式表的大小。

三、存在的问题和拟解决方法

在实现系统的过程中，我遇到了以下问题：

1. 在运行 Neo4j 数据库时，错误的推出导致 Neo4j 数据库无法启动，通过在其 github 库下，找到了问题的解决方案 <https://github.com/neo4j/neo4j/issues/12908>，Neo4j 运行时会将其 pid 保存在该文件中，如果启动时该文件存在，Neo4j 就会认为它已经在运行，从而出错，错误信息为-1。/run 通常会在系统重启时被清除，因为它被挂载为 tmpfs；但在我的容器中，/run 并不是 tmpfs 挂载点，所以它不会被清除，原则上这不应该解决问题，但是，neo4j 对/run 没有写权限，配置更改后那里也没有出现 neo4j.pid 文件，所以似乎它从未创建过 pid 文件。这样就不会再发生 pid 冲突了，但如果我没猜错的话，这意味着只要 neo4j 无法写入，任何目录都可以正常工作，而不仅仅是 /run。通过在 docker-compose.yml 中添加语句解决了问题

2. 数据加载时间过长



图 14 前端展示

在加载球员数据时，由于数据集过大有 1w 条数据，加载时间过长，猜测为语句添加约束过多或者语法不合适[图 15 Cypher 语句]。后期会选择使用专门的工具取进行导入数据集

3. Langchian 的测试版本和 docker 实际运行版本不匹配导致的语句无法运行，后续将选择参考 Langchian 官方文档，修改成符合实际的代码。

4. CSV 文件列名中的特殊字符（例如 Value(£)）导致 Cypher 查询错误。

重命名 CSV 文件的列标题，去掉括号和特殊字符，然后在 Cypher 查询中使用更新后的列名。

5. CSV 文件列名中的特殊字符（例如 Value(£)）导致 Cypher 查询错误。

使用 COALESCE 函数为可能为空的属性提供一个默认值，或者在数据集中填充所有空值。

6. 确定数据属于哪个年份，尤其是当处理多年份数据时。

在节点属性中添加一个明确的年份字段（例如 datasetYear），以区分不同年份的数据。

7. 问题：解析带有货币符号和分隔符的数值字段时出错。

解决方案: 在将数值字段值传递到 Cypher 查询之前, 先在 CSV 中或使用 Cypher 函数 (如 `substring` 和 `replace`) 处理这些符号和分隔符

8. 环境变量问题: 在使用 Docker Compose 时, 环境变量未设置或者错误设置可能导致服务启动不正确。例如, NEO4J 或 API 密钥等变量需要正确配置以保证服务的正常运行。

9. 确保所有必要的环境变量在 `docker-compose.yml` 或 `.env` 文件中正确定义和设置, 以避免运行时错误。

```
LOAD CSV WITH HEADERS FROM 'file:///datas/data1/CLEAN_FIFA23_official_data.txt' AS line FIELDTERMINATOR ','
MERGE (player:Player {
  id: line.ID,
  name: line.Name,
  age: toInteger(line.Age),
  datasetYear: 2023, // 这里直接设置为数据集的年份
  photo: line.Photo,
  nationality: line.Nationality,
  value: toFloat(line.ValueGBP),
  wage: toFloat(line.WageGBP),
  preferredFoot: line.PreferredFoot,
  position: line.Position,
  joined: line.Joined,
  contractValidUntil: toFloat(line.ContractValidUntil),
  height: toFloat(line.Height),
  weight: toFloat(line.Weight),
  yearJoined: toInteger(line.Year_Joined)
})
MERGE (club:Club {name: line.Club})
ON CREATE SET club.logo = line.ClubLogo
MERGE (player)-[:PLAYS_FOR {joined: line.Joined}]->(club)
MERGE (country:Country {name: line.Nationality})
ON CREATE SET country.flag = line.Flag
MERGE (player)-[:COMES_FROM]->(country)
```

图 25 Cypher 语句

附录 3 外文原文

Recent progress in pretraining language models on large textual corpora led to a surge of improvements for downstream NLP tasks. Whilst learning linguistic knowledge, these models may also be storing relational knowledge present in the training data, and may be able to answer queries structured as “fill-in-the-blank” cloze statements. Language models have many advantages over structured knowledge bases: they require no schema engineering, allow practitioners to query about an open class of relations, are easy to extend to more data, and require no human supervision to train. We present an in-depth analysis of the relational knowledge already present (without fine-tuning) in a wide range of state-of-the-art pretrained language models. We find that (i) without fine-tuning, BERT contains relational knowledge competitive with traditional NLP methods that have some access to oracle knowledge, (ii) BERT also does remarkably well on open-domain question answering against a supervised baseline, and (iii) certain types of factual knowledge are learned much more readily than others by standard language model pretraining approaches. The surprisingly strong ability of these models to recall factual knowledge without any fine-tuning demonstrates their potential as unsupervised open-domain QA systems. The code to reproduce our analysis is available at <https://github.com/facebookresearch/LAMA>.

Recently, pretrained high-capacity language models such as ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2018a) have become increasingly important in NLP. They are optimised to either predict the next word in a sequence or some masked word anywhere in a given sequence (e.g. “Dante was born in [Mask] in the year 1265.”). The parameters of these models appear to store vast amounts of linguistic knowledge (Peters et al., 2018b; Goldberg, 2019; Tenney et al., 2019) useful for downstream tasks. This knowledge is usually accessed either by conditioning on latent context representations produced by the original model or by using the original model weights to initialize a task-specific model which is then further fine-tuned. This type of knowledge transfer is crucial for current state-of-the-art results on a wide range of tasks.

Introduction 1 引言

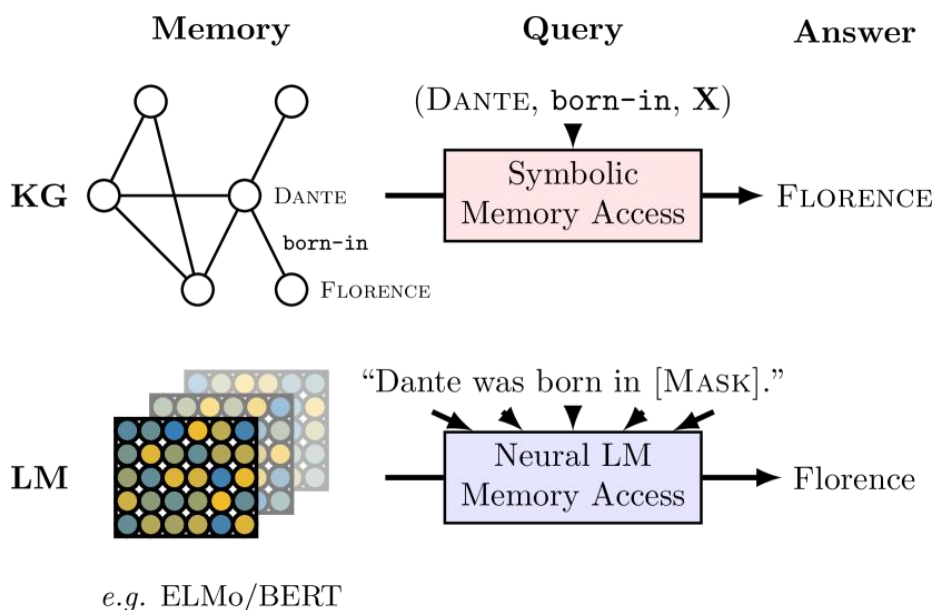


Figure 1: Querying knowledge bases (KB) and language models (LM) for factual knowledge.

In contrast, knowledge bases are effective solutions for accessing annotated gold-standard relational data by enabling queries such as (Dante, born-in,). However, in practice we often need to extract relational data from text or other modalities to populate these knowledge bases. This requires complex NLP pipelines involving entity extraction, coreference resolution, entity linking and relation extraction Surdeanu and Ji (2014)—components that often need supervised data and fixed schemas. Moreover, errors can easily propagate and accumulate throughout the pipeline. Instead, we could attempt to query neural language models for relational data by asking them to fill in masked tokens in sequences like “Dante was born in [Mask]”, as illustrated in Figure 1. In this setting, language models come with various attractive properties: they require no schema engineering, do not need human annotations, and they support an open set of queries.

Given the above qualities of language models as potential representations of relational knowledge, we are interested in the relational knowledge already present in pretrained off-the-shelf language models such as ELMo and BERT. How much relational knowledge do they store? How does this differ for different types of knowledge such as

facts about entities, common sense, and general question answering? How does their performance without fine-tuning compare to symbolic knowledge bases automatically extracted from text? Beyond gathering a better general understanding of these models, we believe that answers to these questions can help us design better unsupervised knowledge representations that could transfer factual and commonsense knowledge reliably to downstream tasks such as commonsense (visual) question answering (Zellers et al., 2018; Talmor et al., 2019) or reinforcement learning (Branavan et al., 2011; Chevalier-Boisvert et al., 2018; Bahdanau et al., 2019; Luketina et al., 2019).

For the purpose of answering the above questions we introduce the LAMA (LAnguage Model Analysis) probe, consisting of a set of knowledge sources, each comprised of a set of facts. We define that a pretrained language model knows a fact (subject, relation, object) such as (Dante, born-in, Florence) if it can successfully predict masked objects in cloze sentences such as “Dante was born in ” expressing that fact. We test for a variety of types of knowledge: relations between entities stored in Wikidata, common sense relations between concepts from ConceptNet, and knowledge necessary to answer natural language questions in SQuAD. In the latter case we manually map a subset of SQuAD questions to cloze sentences.

Our investigation reveals that (i) the largest BERT model from Devlin et al. (2018b) (BERT-large) captures (accurate) relational knowledge comparable to that of a knowledge base extracted with an off-the-shelf relation extractor and an oracle-based entity linker from a corpus known to express the relevant knowledge, (ii) factual knowledge can be recovered surprisingly well from pretrained language models, however, for some relations (particularly -to- relations) performance is very poor, (iii) BERT-large consistently outperforms other language models in recovering factual and commonsense knowledge while at the same time being more robust to the phrasing of a query, and (iv) BERT-large achieves remarkable results for open-domain QA, reaching 57.1% precision@10 compared to 63.5% of a knowledge base constructed using a task-specific supervised relation extraction system .

2 Related Work

Many studies have investigated pretrained word representations, sentence representations, and language models. Existing work focuses on understanding linguistic and semantic properties of word representations or how well pretrained sentence representations and language models transfer linguistic knowledge to downstream tasks. In contrast, our investigation seeks to answer to what extent pretrained language models store factual and commonsense knowledge by comparing them with symbolic knowledge bases populated by traditional relation extraction approaches.

Baroni et al. (2014) present a systematic comparative analysis between neural word representation methods and more traditional count-based distributional semantic methods on lexical semantics tasks like semantic relatedness and concept categorization. They find that neural word representations outperform count-based distributional methods on the majority of the considered tasks. Hill et al. (2015) investigate to what degree word representations capture semantic meaning as measured by similarity between word pairs.

Marvin and Linzen (2018) assess the grammaticality of pretrained language models. Their dataset consists of sentence pairs with a grammatical and an ungrammatical sentence. While a good language model should assign higher probability to the grammatical sentence, they find that LSTMs do not learn syntax well.

Another line of work investigates the ability of pretrained sentence and language models to transfer knowledge to downstream natural language understanding tasks (Wang et al., 2018). While such an analysis sheds light on the transfer-learning abilities of pretrained models for understanding short pieces of text, it provides little insight into whether these models can compete with traditional approaches to representing knowledge like symbolic knowledge bases.

More recently, McCoy et al. (2019) found that for natural language inference, a model based on BERT learns to rely heavily on fallible syntactic heuristics instead of a deeper understanding of the natural language input. Peters et al. (2018b) found that lower layers in ELMo specialize on local syntactic relationships, while higher layers can learn to model long-range relationships. Similarly, Goldberg (2019) found that BERT captures

English syntactic phenomena remarkably well. Tenney et al. (2019) investigate to what extent language models encode sentence structure for different syntactic and semantic phenomena and found that they excel for the former but only provide small improvements for tasks that fall into the latter category. While this provides insights into the linguistic knowledge of language models, it does not provide insights into their factual and commonsense knowledge.

Radford et al. (2018) introduce a pretrained language model based on the Transformer which they termed generative pretraining (GPTv1). The first version of GPT (Radford et al., 2018) has been trained on the Book Corpus (Zhu et al., 2015) containing 7000 books. The closest to our investigation is the work by Radford et al. (2019) which introduces GPTv2 and investigates how well their language model does zero-shot transfer to a range of downstream tasks. They find that GPTv2 achieves an 1 of 55 for answering questions in CoQA (Reddy et al., 2018) and 4.1% accuracy on the Natural Questions dataset (Kwiatkowski et al., 2019), in both cases without making use of annotated question-answer pairs or an information retrieval step. While these results are encouraging and hint at the ability of very large pretrained language models to memorize factual knowledge, the large GPTv2 model has not been made public and the publicly available small version achieves less than 1% on Natural Questions (5.3 times worse than the large model). Thus, we decided to not include GPTv2 in our study. Similarly, we do not include GPTv1 in this study as it uses a limited lower-cased vocabulary, making it incompatible to the way we assess the other language models.

3 The LAMA Probe 4

We introduce the LAMA (LAnguage Model Analysis) probe to test the factual and commonsense knowledge in language models. It provides a set of knowledge sources which are composed of a corpus of facts. Facts are either subject-relation-object triples or question-answer pairs. Each fact is converted into a cloze statement which is used to query the language model for a missing token. We evaluate each model based on how highly it

ranks the ground truth token against every other word in a fixed candidate vocabulary. This is similar to ranking-based metrics from the knowledge base completion literature (Bordes et al., 2013; Nickel et al., 2016). Our assumption is that models which rank ground truth tokens high for these cloze statements have more factual knowledge. We discuss each step in detail next and provide considerations on the probe below.

3.1 Knowledge Sources

To assess the different language models in Section 2, we cover a variety of sources of factual and commonsense knowledge. For each source, we describe the origin of fact triples (or question-answer pairs), how we transform them into cloze templates, and to what extent aligned texts exist in Wikipedia that are known to express a particular fact. We use the latter information in supervised baselines that extract knowledge representations directly from the aligned text.

4.1.1 Google-RE

The Google-RE contains $\sim 60K$ facts manually extracted from Wikipedia. It covers five relations but we consider only three of them, namely “place of birth”, “date of birth” and “place of death”. We exclude the other two because they contain mainly multi-tokens objects that are not supported in our evaluation. We manually define a template for each considered relation, e.g., “[S] was born in [O]” for “place of birth”. Each fact in the Google-RE dataset is, by design, manually aligned to a short piece of Wikipedia text supporting it.

4.1.2 T-REx

The T-REx knowledge source is a subset of Wikidata triples. It is derived from the T-REx dataset Elsahar et al. (2018) and is much larger than Google-RE with a broader set of relations. We consider 41 Wikidata relations and subsample at most 1000 facts per relation. As with the Google-RE corpus, we manually define a template for each relation (see Table 3 for some examples). In contrast to the Google-RE knowledge source, T-REx facts were automatically aligned to Wikipedia and hence this alignment can be noisy. However, Elsahar et al. (2018) report an accuracy of 97.8% for the alignment technique

over a test set.

4.1.3 ConceptNet

ConceptNet Speer and Havasi (2012) is a multi-lingual knowledge base, initially built on top of Open Mind Common Sense (OMCS) sentences. OMCS represents commonsense relationships between words and/or phrases. We consider facts from the English part of ConceptNet that have single-token objects covering 16 relations. For these ConceptNet triples, we find the OMCS sentence that contains both the subject and the object. We then mask the object within the sentence and use the sentence as template for querying language models. If there are several sentences for a triple, we pick one at random. Note that for this knowledge source there is no explicit alignment of facts to Wikipedia sentences.

4.1.4 SQuAD

SQuAD Rajpurkar et al. (2016) is a popular question answering dataset. We select a subset of 305 context-insensitive questions from the SQuAD development set with single token answers. We manually create cloze-style questions from these questions, e.g., rewriting “Who developed the theory of relativity?” as “The theory of relativity was developed by ”. For each question and answer pair, we know that the corresponding fact is expressed in Wikipedia since this is how SQuAD was created.

4.2 Models

We consider the following pretrained case-sensitive language models in our study (see Table 1): fairseq-fconv (Fs), Transformer-XL large (Txl), ELMo original (Eb), ELMo 5.5B (E5B), BERT-base (Bb) and BERT-large (Bl). We use the natural way of generating tokens for each model by following the definition of the training objective function.

Assume we want to compute the generation for the token at position i . For unidirectional language models, we use the network output ($i-1$) just before the token to produce the output layer softmax. For ELMo we consider the output just before ($i \rightarrow i-1$) for the forward direction and just after ($i \leftarrow i+1$) for the backward direction. Following the loss definition in (Peters et al., 2018a), we average forward and backward probabilities

from the corresponding softmax layers. For BERT, we mask the token at position i , and we feed the output vector corresponding to the masked token (x_i) into the softmax layer. To allow a fair comparison, we let models generate over a unified vocabulary, which is the intersection of the vocabularies for all considered models ($\sim 21K$ case-sensitive tokens).

4.3 Baselines

To compare language models to canonical ways of using off-the-shelf systems for extracting symbolic knowledge and answering questions, we consider the following baselines.

Freq: For a subject and relation pair, this baseline ranks words based on how frequently they appear as objects for the given relation in the test data. It indicates the upper bound performance of a model that always predicts the same objects for a particular relation.

RE: For the relation-based knowledge sources, we consider the pretrained Relation Extraction (RE) model of Sorokin and Gurevych (2017). This model was trained on a subcorpus of Wikipedia annotated with Wikidata relations. It extracts relation triples from a given sentence using an LSTM-based encoder and an attention mechanism. Based on the alignment information from the knowledge sources, we provide the relation extractor with the sentences known to express the test facts. Using these datasets, RE constructs a knowledge graph of triples. At test time, we query this graph by finding the subject entity and then rank all objects in the correct relation based on the confidence scores returned by RE. We consider two versions of this procedure that differ in how the entity linking is implemented: **REn** makes use of a naïve entity linking solution based on exact string matching, while **REo** uses an oracle for entity linking in addition to string matching. In other words, assume we query for the object o of a test subject-relation fact expressed in a sentence s . If RE has extracted any triple from that sentence s will be linked to o and o to s . In practice, this means RE can return the correct solution o if any relation instance of the right type was extracted from s , regardless of whether it has a wrong subject or object.

DrQA:

Chen et al. (2017) introduce DrQA, a popular system for open-domain question answering. DrQA predicts answers to natural language questions using a two step pipeline. First, a TF/IDF information retrieval step is used to find relevant articles from a large store of documents (e.g. Wikipedia). On the retrieved top ,articles, a neural reading comprehension model then extracts answers. To avoid giving the language models a competitive advantage, we constrain the predictions of DrQA to single-token answers.

4.4Metrics

We consider rank-based metrics and compute results per relation along with mean values across all relations. To account for multiple valid objects for a subject-relation pair (i.e., for N-M relations), we follow Bordes et al. (2013) and remove from the candidates when ranking at test time all other valid objects in the training data other than the one we test. We use the mean precision at k ($P@k$). For a given fact, this value is 1 if the object is ranked among the top k results, and 0 otherwise.

4.5Considerations

There are several important design decisions we made when creating the LAMA probe. Below we give more detailed justifications for these decisions.

Manually Defined Templates

For each relation we manually define a template that queries for the object slot in that relation. One can expect that the choice of templates has an impact on the results, and this is indeed the case: for some relations we find both worse and better ways to query for the same information (with respect to a given model) by using an alternate template. We argue that this means we are measuring a lower bound for what language models know. We make this argument by analogy with traditional knowledge bases: they only have a single way of querying knowledge for a specific relation, namely by using the relation id of that relation, and this way is used to measure their accuracy. For example, if the relation

ID is works-For and the user asks for is-working-for, the accuracy of the KG would be 0.

Single Token

We only consider single token objects as our prediction targets. The reason we include this limitation is that multi-token decoding adds a number of additional tuneable parameters (beam size, candidate scoring weights, length normalization, n-gram repetition penalties, etc.) that obscure the knowledge we are trying to measure. Moreover, well-calibrated multi-token generation is still an active research area, particularly for bidirectional models (see e.g. Welleck et al. (2019)).

Object Slots

We choose to only query object slots in triples, as opposed to subject or relation slots. By including reverse relations (e.g. contains and contained-by) we can also query subject slots. We do not query relation slots for two reasons. First, surface form realisations of relations will span several tokens, and as we discussed above, this poses a technical challenge that is not in the scope of this work. Second, even if we could easily predict multi-token phrases, relations can generally be expressed with many different wordings, making it unclear what the gold standard pattern for a relation should be, and how to measure accuracy in this context.

Intersection of Vocabularies

The models that we considered are trained with different vocabularies. For instance, ELMo uses a list of ~ 800K tokens while BERT considers only ~ 30K tokens. The size of the vocabulary can influence the performance of a model for the LAMA probe. Specifically, the larger the vocabulary the harder it would be to rank the gold token at the top. For this reason we considered a common vocabulary of ~ 21K case-sensitive tokens

that are obtained from the intersection of the vocabularies for all considered models. To allow a fair comparison, we let every model rank only tokens in this joint vocabulary.

5 Results

We summarize the main results in Table 2, which shows the mean precision at one (P@1) for the different models across the set of corpora considered. In the remainder of this section, we discuss the results for each corpus in detail.

Google-RE

We query the LMs using a standard cloze template for each relation. The base and large versions of BERT both outperform all other models by a substantial margin. Furthermore, they obtain a 2.2 and 2.9 respective average accuracy improvement over the oracle-based RE baseline. This is particularly surprising given that with the gold-aligned Google-RE source we know for certain that the oracle RE baseline has seen at least one sentence expressing each test fact. Moreover, the RE baseline was given substantial help through an entity linking oracle.

It is worth pointing out that while BERT-large does better, this does not mean it does so for the right reasons. Although the aligned Google-RE sentences are likely in its training set (as they are part of Wikipedia and BERT has been trained on Wikipedia), it might not “understand” them to produce these results. Instead, it could have learned associations of objects with subjects from co-occurrence patterns.

T-REx

The knowledge source derived from Google-RE contains relatively few facts and only three relations. Hence, we perform experiments on the larger set of facts and relations in T-REx. We find that results are generally consistent with Google-RE. Again, the

performance of BERT in retrieving factual knowledge are close to the performance obtained by automatically building a knowledge base with an off-the-shelf relation extraction system and oracle-based entity linking. Broken down by relation type, the performance of BERT is very high for 1-to-1 relations (e.g., capital of) and low for N-to-M relations.

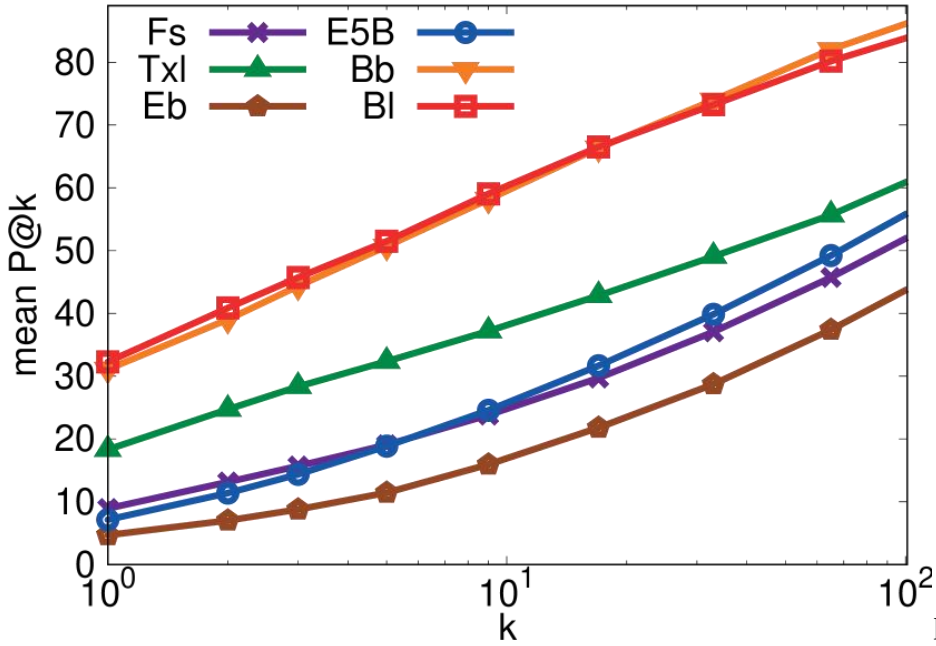


Figure 2: Mean

P@k curve for T-REx varying k. Base-10 log scale for X axis.

Note that a downstream model could learn to make use of knowledge in the output representations of a language model even if the correct answer is not ranked first but high enough (i.e. a hint about the correct answer can be extracted from the output representation). Figure 2 shows the mean P@k curves for the considered models. For BERT, the correct object is ranked among the top ten in around 60% of the cases and among the top 100 in 80% of the cases.

To further investigate why BERT achieves such strong results, we compute the Pearson correlation coefficient between the P@1 and a set of metrics that we report in Figure 8. We notice, for instance, that the number of times an object is mentioned in the training data positively correlates with performance while the same is not true for the subject of a relation. Furthermore, the log probability of a prediction is strongly positively

correlated with $P@1$. Thus, when BERT has a high confidence in its prediction, it is often correct. Performance is also positively correlated with the cosine similarity between subject and object vectors, and slightly with the number of tokens in the subject.

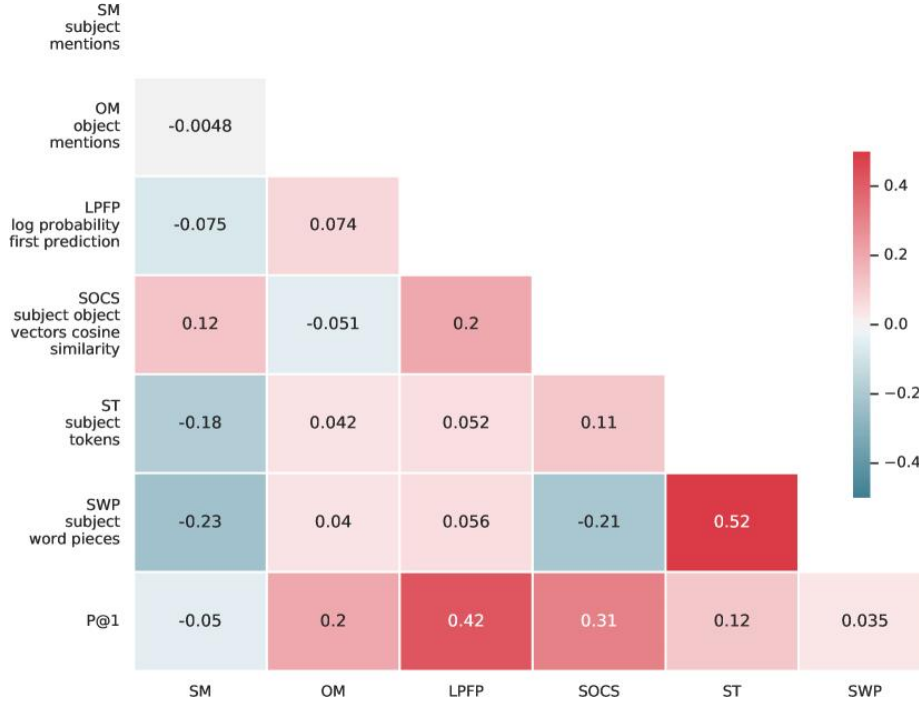


Figure 3: Pearson correlation coefficient for the $P@1$ of the BERT-large model on T-REx and a set of metrics: SM and OM refer to the number of times a subject and an object are mentioned in the BERT training corpus⁷ respectively; LPFP is the log probability score associated with the first prediction; SOCS is the cosine similarity between subject and object vectors (we use spaCy⁸); ST and SWP are the number of tokens in the subject with a standard tokenization and the BERT WordPiece tokenization respectively.

To understand how the performance of a pretrained language model varies with different ways of querying for a particular fact, we analyze a maximum of 100 random facts per relation for which we randomly select 10 aligned sentences in Wikipedia from T-REx.⁹ We exclude all facts with less than 10 alignments.

⁹ In each of the sentences, we mask the object of the fact, and ask the model to predict it. For several of our language models this also tests their ability to memorize and recall sentences from the training data since as the models have been trained on Wikipedia (see Table 1).

Figure 4 shows the average distribution of the rank for ten queries per fact. The two BERT models and ELMo 5.5B exhibit the lowest variability while ranking the correct object close to the top on average. Surprisingly, the performance of ELMo original is not far from BERT, even though this model did not see Wikipedia during training. Fairseq-fconv and Transformer-XL experience a higher variability in their predictions. Note that BERT and ELMo 5.5B have been trained on a larger portion of Wikipedia than fairseq-fconv and Transformer-XL and may have seen more sentences containing the test queries during training.

ConceptNet 概念网

The results on the ConceptNet corpus are in line with those reported for retrieving factual knowledge in Google-RE and T-REx. The BERT-large model consistently achieves the best performance, and it is able to retrieve commonsense knowledge at a similar level to factual knowledge. The lower half of Table 3 shows generations by BERT-large for randomly sampled examples. Some of the concepts generated by the language models are surprisingly reasonable in addition to being syntactically correct.

SQuAD

Next we evaluate our system on open-domain cloze-style question answering and compare against the supervised DrQA model. Table 2 shows a performance gap between BERT-large and the DrQA open-domain QA system on our cloze SQuAD task. Again, note that the pretrained language model is completely unsupervised, it is not fine-tuned, and it has no access to a dedicated information retrieval system. Moreover, when comparing DrQA and BERT-large in terms of $P@10$, we find that gap is remarkably small (57.1 for BERT-large and 63.5 for DrQA).

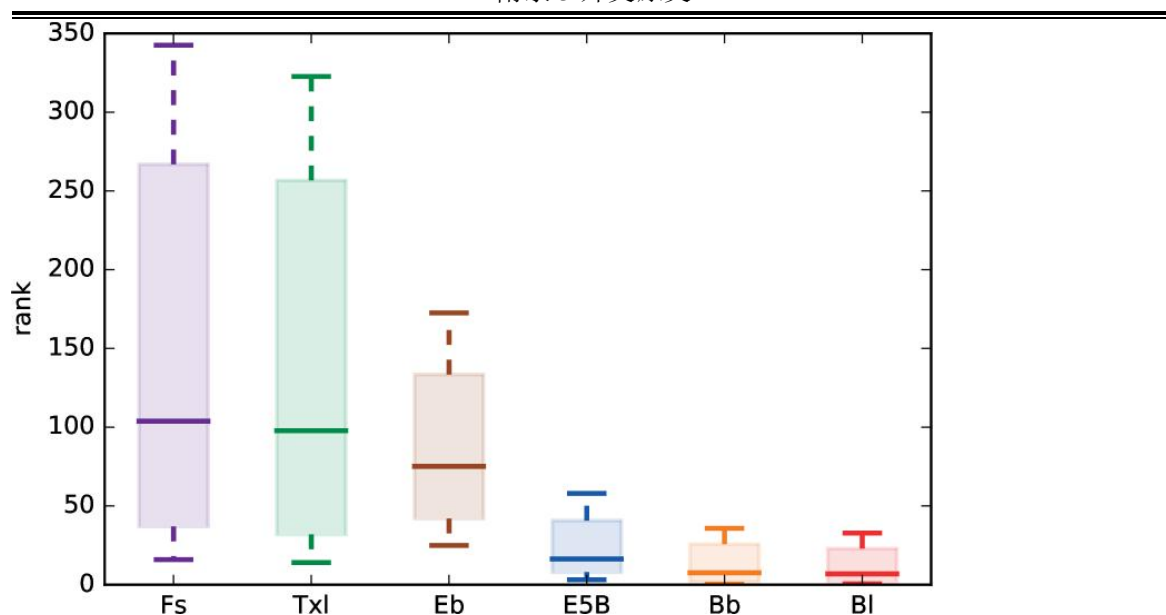


Figure 4: Average rank distribution for 10 different mentions of 100 random facts per relation in T-REx.

ELMo 5.5B and both variants of BERT are least sensitive to the framing of the query but also are the most likely to have seen the query sentence during training.

6 Discussion and Conclusion

We presented a systematic analysis of the factual and commonsense knowledge in publicly available pretrained language models as is and found that BERT-large is able to recall such knowledge better than its competitors and at a level remarkably competitive with non-neural and supervised alternatives. Note that we did not compare the ability of the corresponding architectures and objectives to capture knowledge in a given body of text but rather focused on the knowledge present in the weights of existing pretrained models that are being used as starting points for many researchers’ work. Understanding which aspects of data our commonly-used models and learning algorithms are capturing is a crucial field of research and this paper complements the many studies focused on the learned linguistic properties of the data.

We found that it is non-trivial to extract a knowledge base from text that performs on par to directly using pretrained BERT-large. This is despite providing our relation extraction baseline with only data that is likely expressing target facts, thus reducing potential for false negatives, as well as using a generous entity-linking oracle. We

suspected BERT might have an advantage due to the larger amount of data it has processed, so we added Wikitext-103 as additional data to the relation extraction system and observed no significant change in performance. This suggests that while relation extraction performance might be difficult to improve with more data, language models trained on ever growing corpora might become a viable alternative to traditional knowledge bases extracted from text in the future.

In addition to testing future pretrained language models using the LAMA probe, we are interested in quantifying the variance of recalling factual knowledge with respect to varying natural language templates. Moreover, assessing multi-token answers remains an open challenge for our evaluation setup.

附录 4 外文翻译

最近，在大型文本语料库上对语言模型进行预训练方面取得的进展，使下游的 NLP 任务有了突飞猛进的改进。在学习语言知识的同时，这些模型可能还存储了训练数据中的关系知识，并能够回答结构化为 "填空" 式语句的查询。与结构化知识库相比，语言模型有很多优势：它们不需要模式工程，允许从业人员查询开放的关系类别，易于扩展到更多数据，而且不需要人工监督训练。我们对各种最先进的预训练语言模型中已有的关系知识（未经微调）进行了深入分析。我们发现：(i) 在不进行微调的情况下，BERT 所包含的关系知识与传统的 NLP 方法相比具有一定的竞争力，因为传统的 NLP 方法在一定程度上可以获取甲骨文知识；(ii) BERT 在开放域问题解答方面的表现也非常出色，与有监督的基线相比毫不逊色；(iii) 通过标准的语言模型预训练方法，某些类型的事实知识比其他类型的知识更容易被学习到。这些模型在不进行任何微调的情况下召回事实知识的能力之强令人吃惊，这证明了它们作为无监督开放域质量保证系统的潜力。重现我们的分析的代码可在 <https://github.com/facebookresearch/LAMA> 上获取。

最近，ELMo (Peters 等人, 2018a) 和 BERT (Devlin 等人, 2018a) 等预训练的高容量语言模型在 NLP 中变得越来越重要。这些模型经过优化，可以预测序列中的下一个单词，也可以预测给定序列中任意位置的某个掩码单词（例如 "但丁于 1265 年出生在 [掩码]"）。这些模型的参数似乎存储了大量对下游任务有用的语言知识 (Peters 等人, 2018b; Goldberg, 2019; Tenney 等人, 2019)。这些知识通常是通过原始模型产生的潜在语境表征进行调节，或者使用原始模型的权重来初始化特定任务模型，然后对其进行进一步微调。这种类型的知识转移对于当前在各种任务中取得最先进的结果至关重要。

与此相反，知识库通过支持诸如 (但丁, born-in,) 这样的查询，成为访问附有注释的黄金标准关系数据的有效解决方案。然而，在实践中，我们往往需要从文本或其他模式中提取关系数据来填充这些知识库。这需要复杂的 NLP 管道，其中涉及实体提取、核心参照解析、实体链接和关系提取 Surdeanu 和 Ji (2014 年) -- 这些组件通常需要有监督的数据和固定的模式。此外，错误很容易在整个管道中传

播和积累。相反，我们可以尝试让神经语言模型填写 "但丁出生于 [掩码]"等序列中的掩码标记，从而查询关系数据，如图 1 所示。在这种情况下，语言模型具有各种吸引人的特性：它们不需要模式工程，不需要人工注释，而且支持开放式查询。

引言 1

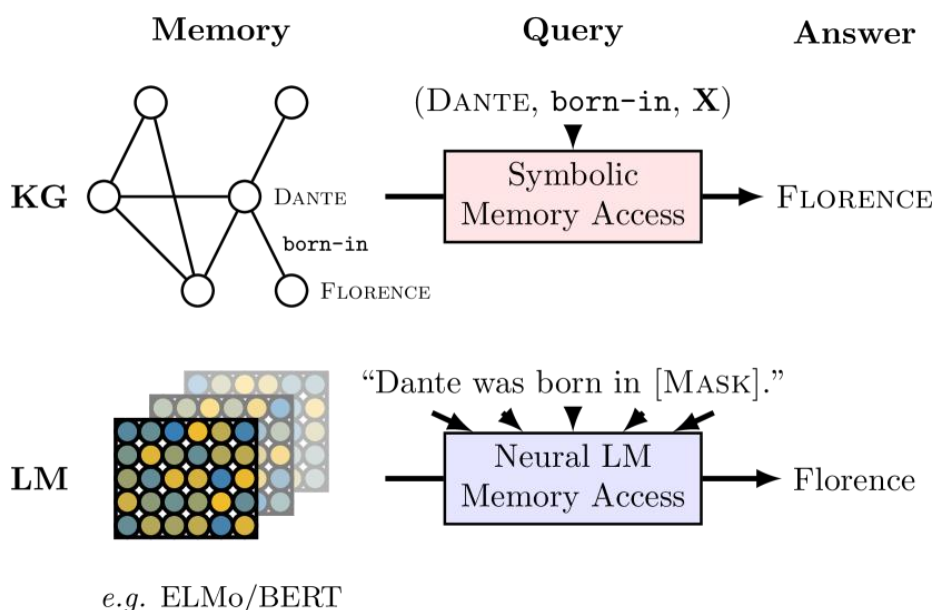


图 1: 查询事实知识的知识库 (KB) 和语言模型 (LM)。

鉴于语言模型作为关系知识潜在表征的上述特性，我们对 ELMo 和 BERT 等预训练现成语言模型中已经存在的关系知识很感兴趣。它们存储了多少关系知识？对于不同类型的知识（如实体事实、常识和一般问题解答），关系知识有什么不同？与自动从文本中提取的符号知识库相比，它们在不进行微调的情况下的性能如何？除了加深对这些模型的一般理解之外，我们相信，对这些问题的回答可以帮助我们设计出更好的无监督知识表征，从而将事实和常识知识可靠地转移到下游任务中，例如常识（视觉）问题解答（Zellers 等人，2018 年；Talmor 等人，2019 年）或强化学习（Branavan 等人，2011 年；Chevalier-Boisvert 等人，2018 年；Bahdanau 等人，2019 年；Luketina 等人，2019 年）。

为了回答上述问题，我们引入了 LAMA（语言模型分析）探针，它由一组知识源组成，每个知识源由一组事实组成。我们的定义是，如果预训练语言模型能够成功预测 "但丁出生于 "等表达事实的掐头去尾句子中的遮蔽对象，那么它就知道某个事实（主语，relation，宾语），例如（但丁，born-in，佛罗伦萨）。我们测试了

多种类型的知识：存储在 Wikidata 中的实体之间的关系、来自 ConceptNet 的概念之间的常识关系，以及回答 SQuAD 中自然语言问题所需的知识。在后一种情况下，我们手动将 SQuAD 问题的一个子集映射为掐头去尾的句子。

我们的研究发现：(i) Devlin 等人的最大 BERT 模型 (BERT-large) 捕获的（准确）关系知识与使用现成关系提取器和基于实体链接器从语料库中提取的知识库相当。(2018b) (BERT-large) 捕捉到的（准确的）关系知识与使用现成的关系提取器和基于 oracle 的实体链接器从已知表达相关知识的语料库中提取的知识库的知识相当，(ii) 然而，事实知识可以从预训练的语言模型中得到令人惊讶的恢复、(iii) BERT-large 在恢复事实知识和常识性知识方面始终优于其他语言模型，同时对查询措辞的鲁棒性也更强；(iv) BERT-large 在开放域质量保证方面取得了显著的成果，达到了 57.1%@10，而使用特定任务监督关系提取系统构建的知识库的精确度为 63.5%。

2 相关工作

许多研究都对预训练的单词表征、句子表征和语言模型进行了调查。现有研究的重点是了解单词表征的语言和语义属性，或者预训练的句子表征和语言模型如何将语言知识转移到下游任务中。与此相反，我们的研究试图通过将预训练语言模型与传统关系提取方法填充的符号知识库进行比较，来回答预训练语言模型在多大程度上存储了事实和常识知识。

Baroni 等人 (2014 年) 在语义相关性和概念分类等词汇语义学任务中，对神经词表示方法和更传统的基于计数的分布式语义方法进行了系统的比较分析。他们发现，在所考虑的大多数任务中，神经词表示法都优于基于计数的分布式方法。Hill 等人 (2015 年) 研究了词表征在多大程度上捕捉到了以词对之间的相似性来衡量的语义。

Marvin 和 Linzen (2018) 评估了预训练语言模型的语法性。他们的数据集由包含一个语法句和一个非语法句的句对组成。虽然一个好的语言模型应该为语法句子分配更高的概率，但他们发现 LSTM 的语法学习效果并不好。

另一项工作研究了预训练句子和语言模型将知识迁移到下游自然语言理解任务的能力 (Wang 等人, 2018 年)。虽然这种分析揭示了预训练模型在理解短小文本

时的迁移学习能力，但对于这些模型是否能与符号知识库等传统知识表示方法竞争，却没有提供什么深入的见解。

最近，McCoy 等人（2019 年）发现，对于自然语言推理，基于 BERT 的模型学会严重依赖易错的句法启发式，而不是对自然语言输入的深入理解。Peters 等人（2018b）发现，ELMo 的低层专门处理局部句法关系，而高层则可以学习建立长程关系模型。同样，Goldberg（2019）发现，BERT 能很好地捕捉英语句法现象。Tenney 等人（2019）研究了语言模型在多大程度上对不同句法和语义现象的句子结构进行了编码，发现它们在前者方面表现出色，但在属于后者的任务方面仅有微小改进。虽然这有助于深入了解语言模型的语言知识，但却无法深入了解它们的事实和常识知识。

Radford 等人（2018 年）介绍了一种基于 Transformer 的预训练语言模型，他们称之为生成式预训练（GPTv1）。GPT 的第一个版本（Radford 等人，2018 年）是在包含 7000 本图书的图书语料库（Zhu 等人，2015 年）上训练出来的。与我们的研究最接近的是 Radford 等人的研究（2019 年），他们引入了 GPTv2，并研究了其语言模型在一系列下游任务中的零点转移效果。他们发现，GPTv2 在回答 CoQA（Reddy 等人，2018 年）中的问题时达到了 55 的 1 精度，在自然问题数据集（Kwiatkowski 等人，2019 年）上达到了 4.1% 精度，在这两种情况下，都没有使用注释的问答对或信息检索步骤。虽然这些结果令人鼓舞，并暗示了超大型预训练语言模型记忆事实知识的能力，但大型 GPTv2 模型尚未公开，而公开的小型版本在 Natural Questions 上的成绩低于 1%（比大型模型差 5.3 倍）。因此，我们决定不将 GPTv2 纳入我们的研究。同样，我们也没有将 GPTv1 纳入本研究，因为它使用了有限的小写词汇，与我们评估其他语言模型的方法不兼容。

3 LAMA 探测器

我们引入了 LAMA（语言模型分析）探针来测试语言模型中的事实和常识知识。它提供了一组由事实语料库组成的知识源。事实可以是主题-关系-对象三元组，也可以是问题-答案对。每个事实都会被转换成一个掐头去尾的语句，用来查询语言模型中缺失的标记。我们评估每个模型的依据是，根据固定候选词库中的每个其他单词，该模型对基本事实标记的排序有多高。这与知识库补全文献中基于排名的指标类似

(Bordes 等人, 2013 年; Nickel 等人, 2016 年)。我们的假设是, 在这些掐头去尾的语句中, 地面真实标记排名靠前的模型拥有更多的事实知识。接下来我们将详细讨论每个步骤, 并在下文中提供有关探究的考虑因素。

3.1 知识来源

为了评估第 2 节中的不同语言模型, 我们涵盖了各种事实和常识知识来源。对于每个来源, 我们都会描述事实三元组 (或问答对) 的来源、如何将其转换为掐词模板, 以及维基百科中已知表达特定事实的对齐文本的存在程度。我们在监督基线中使用后一种信息, 直接从对齐文本中提取知识表征。

3.1.1 谷歌-RE

包含从维基百科手动提取的 ~ 60 K 个事实。它涵盖五种关系, 但我们只考虑其中三种, 即 "出生地"、"出生日期" 和 "死亡地点"。我们排除了另外两个关系, 因为它们主要包含多标记对象, 而我们的评估不支持多标记对象。我们为每个考虑的关系手动定义了一个模板, 例如, "出生地" 的模板是 "[S] 出生于 [O]"。根据设计, Google-RE 数据集中的每个事实都是与维基百科中支持该事实的简短文本人工对齐的。

4.1.2 T-REx

T-REx 知识源是维基数据三元组的一个子集。它源自 T-REx 数据集 Elshahar 等人 (2018 年), 比 Google-RE 大得多, 关系集也更广泛。我们考虑了 41 个维基数据关系, 每个关系最多抽取 1000 个事实。与 Google-RE 语料库一样, 我们为每个关系手动定义了一个模板 (部分示例见表 3)。与 Google-RE 知识源不同的是, T-REx 的事实是自动与维基百科对齐的, 因此这种对齐可能会产生噪音。不过, Elshahar 等人 (2018 年) 报告称, 该对齐技术在测试集上的准确率为 97.8%。

4.1.3 概念网络

ConceptNet Speer 和 Havasi (2012 年) 是一个多语言知识库, 最初建立在开放思维常识 (OMCS) 句子之上。OMCS 表示单词和/或短语之间的常识关系。我们考虑了 ConceptNet 英语部分的事实, 这些事实的单标记对象涵盖了 16 种关系。对于这些 ConceptNet 三元组, 我们会找到同时包含主语和宾语的 OMCS 句子。然后, 我们屏蔽句子中的宾语, 并将该句子作为查询语言模型的模板。如果一个三元组有

多个句子，我们会随机挑选一个。请注意，对于这种知识源，没有将事实与维基百科句子明确对齐。

4.1.4 SQuAD

SQuAD Rajpurkar 等人（2016 年）是一个流行的问题解答数据集。我们从 SQuAD 开发集中选取了 305 个上下文敏感问题的子集，这些问题的答案都是单一标记。我们从这些问题中手动创建了掐头去尾式的问题，例如，将 "谁提出了相对论？" 改写为 "相对论是由"。对于每一对问答，我们都知道维基百科中表达了相应的事实，因为 SQuAD 就是这样创建的。

4.2 模型

我们在研究中使用了以下经过预训练的大小写敏感语言模型（见表 1）：Fairseq-fconv (Fs)、Transformer-XL large (Txl)、ELMo original (Eb)、ELMo 5.5B (E5B)、BERT-base (Bb) 和 BERT-large (Bl)。我们根据训练目标函数的定义，采用自然方式为每个模型生成标记。

假设我们要计算位于 位置的标记的生成。对于单向语言模型，我们使用该标记前的网络输出来生成输出层 softmax。对于 ELMo，我们考虑的是前向输出和后向输出。根据（Peters 等，2018a）中的损失定义，我们对相应 softmax 层的前向和后向概率进行平均。对于 BERT，我们屏蔽了位置 上的标记，并将屏蔽标记对应的输出向量输入到 softmax 层。为了进行公平的比较，我们让模型在一个统一的词汇表上生成，该词汇表是所有模型词汇表的交集。

4.3 基准

为了将语言模型与使用现成系统提取符号知识和回答问题的典型方法进行比较，我们考虑了以下基线。

频率（Freq）：对于主题和关系对，该基线根据词语在测试数据中作为给定关系的对象出现的频率对词语进行排序。它表示一个模型的性能上限，该模型总是预测特定关系中的相同对象。

关系提取对于基于关系的知识源，我们考虑使用 Sorokin 和 Gurevych（2017 年）的预训练关系提取（RE）模型。该模型是在注有维基数据关系的维基百科子语料库

上训练的。它使用基于 LSTM 的编码器和注意力机制从给定句子中提取关系三元组。根据知识源的对齐信息，我们为关系提取器提供了已知表达测试事实的句子。利用这些数据集，RE 构建了一个三元组知识图谱。测试时，我们通过查找主题实体来查询该图，然后根据 RE 返回的置信度分数对正确关系中的所有对象进行排序。我们考虑了这一程序的两个版本，它们在实体链接的实现方式上有所不同：RE_n使用的是基于精确字符串匹配的简单实体链接解决方案，而 RE_o除了使用字符串匹配外，还使用了实体链接神谕。换句话说，假设我们查询句子中表达的测试主题相关事实的对象。如果 RE 已从该句子中提取了任何三元，则链接到，而将链接到。在实践中，这意味着如果从中提取了正确类型的关系实例，则 RE 可以返回正确的解决方案，而不管它的主语或宾语是否错误。

DrQA: Chen 等人（2017 年）介绍了 DrQA，这是一个用于开放领域问题解答的流行系统。DrQA 通过两步管道预测自然语言问题的答案。首先，采用 TF/IDF 信息检索步骤，从大型文档库（如维基百科）中查找相关文章。然后，神经阅读理解模型在检索到的顶级文章中提取答案。为了避免给语言模型带来竞争优势，我们将 DrQA 的预测限制在单标记答案上。

4.5 考虑因素

在创建 LAMA 探头时，我们做出了几个重要的设计决定。下面我们将详细说明这些决定的理由。

手动定义模板

我们为每种关系手动定义了一个模板，用于查询该关系中的对象插槽。我们可以预料到模板的选择会对结果产生影响，而事实也确实如此：对于某些关系，我们发现使用另一种模板查询相同信息（相对于给定的模型）的方法既有更差的，也有更好的。我们认为，这意味着我们测量的是语言模型已知信息的下限。我们通过类比传统知识库来提出这一论点：传统知识库只有一种查询特定关系知识的方法，即使用该关系的关系 ID，而这种方法被用来衡量知识库的准确性。例如，如果关系 ID 是 works-For，而用户询问的是 is-working-for，那么知识库的准确性就是 0。

单一令牌

我们只将单标记对象作为预测目标。我们之所以做出这样的限制，是因为多标记解码增加了许多额外的可调整参数（波束大小、候选评分权重、长度归一化、n-gram 重复惩罚等），这些参数掩盖了我们试图测量的知识。此外，校准良好的多标记词生成仍是一个活跃的研究领域，尤其是对于双向模型（参见 Welleck 等人（2019））。

对象插槽

我们选择只查询三元组中的对象插槽，而不查询主题或关系插槽。通过包含反向关系（例如 `contains` 和 `contained-by`），我们也可以查询主语槽。我们不查询关系槽有两个原因。首先，关系的表面形式实现将跨越多个标记，正如我们在上文所讨论的，这将带来技术上的挑战，不在本文的研究范围之内。其次，即使我们可以很容易地预测多标记短语，关系一般也可以用许多不同的措辞来表达，这就不清楚关系的黄金标准模式应该是什么，也不清楚在这种情况下如何衡量准确性。

我们所考虑的模型是用不同的词汇表训练出来的。例如，ELMo 使用了一个包含 ~ 800K 词组的列表，而 BERT 只考虑了 ~ 30K 词组。词汇量的大小会影响 LAMA 探针模型的性能。具体来说，词汇量越大，黄金标记就越难排在前面。因此，我们考虑了一个由 ~ 21K 大小写敏感标记组成的通用词汇表，该词汇表是从所有考虑过的模型的词汇表的交集中获得的。为了进行公平的比较，我们让每个模型只对这个联合词汇表中的标记进行排名。

5 结果

我们在表 2 中总结了主要结果，该表显示了在所考虑的语料集中，不同模型的平均精确度（P@1）。在本节的其余部分，我们将详细讨论每个语料库的结果。

我们使用每个关系的标准掐头去尾模板来查询 LM。BERT 的基本版本和大型版本都大大优于所有其他模型。此外，与基于甲骨文的 RE 基线相比，它们的平均准确率分别提高了 2.2 和 2.9。这一点尤其令人惊讶，因为我们可以确定，使用黄金对齐的 Google-RE 源，甲骨文 RE 基线至少看到过一个表达每个测试事实的句子。

此外，实体链接甲骨文还为 RE 基线提供了大量帮助。

值得指出的是，虽然 BERT-large 做得更好，但这并不意味着它这样做的原因是正确的。虽然对齐的 Google-RE 句子很可能在它的训练集中（因为它们是维基百科的一部分，而 BERT 也曾 在 维 基 百 科 上 接 受 过 训 练），但它可能并不“理解”这些句子，因而无法得出这些结果。相反，它可能是从共现模式中学习了对象与主题的关联。

T-REx

从 Google-RE 派生的知识源包含相对较少的事实和仅有的三种关系。因此，我们对 T-REx 中更多的事实和关系集进行了实验。我们发现结果与 Google-RE 基本一致。同样，BERT 在检索事实知识方面的性能也接近于使用现成的关系提取系统和基于甲骨文的实体链接自动构建知识库所获得的性能。按关系类型细分，BERT 在处理 1 对 1 关系（如 capital of）时性能非常高，而在处理 N 对 M 关系时性能较低。

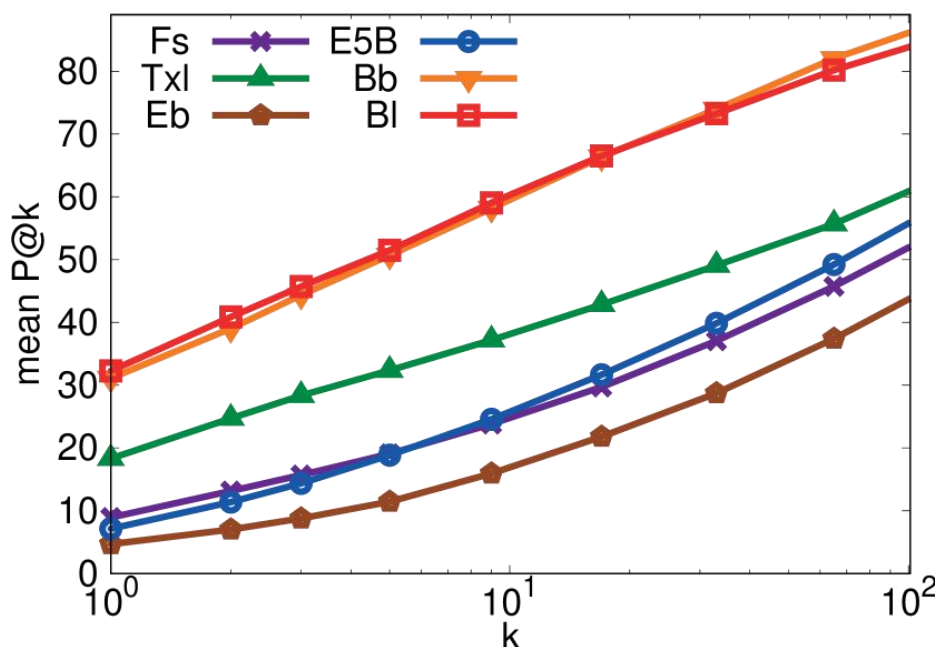


图 2: T-REx 变化 k 的平均 P@k 曲线。

需要注意的是，下游模型可以学习利用语言模型输出表示中的知识，即使正确答案没有排在第一位，但也足够靠前（即可以从输出表示中提取关于正确答案的提

示)。图 2 显示了所考虑模型的平均 $P@k$ 曲线。就 BERT 而言,在大约 60% 的情况下,正确答案被排在前 10 位,而在 80% 的情况下,正确答案被排在前 100 位。

为了进一步探究 BERT 取得如此优异成绩的原因,我们计算了 $P@1$ 与一系列指标之间的皮尔逊相关系数,如图 8 所示。例如,我们注意到,在训练数据中,对象被提及的次数与性能呈正相关,而关系的主体则不然。此外,预测的对数概率与 $P@1$ 呈强正相关。因此,当 BERT 对自己的预测有很高的信心时,它往往是正确的。性能还与主语和宾语向量之间的余弦相似度呈正相关,并与主语中的标记数略有相关。

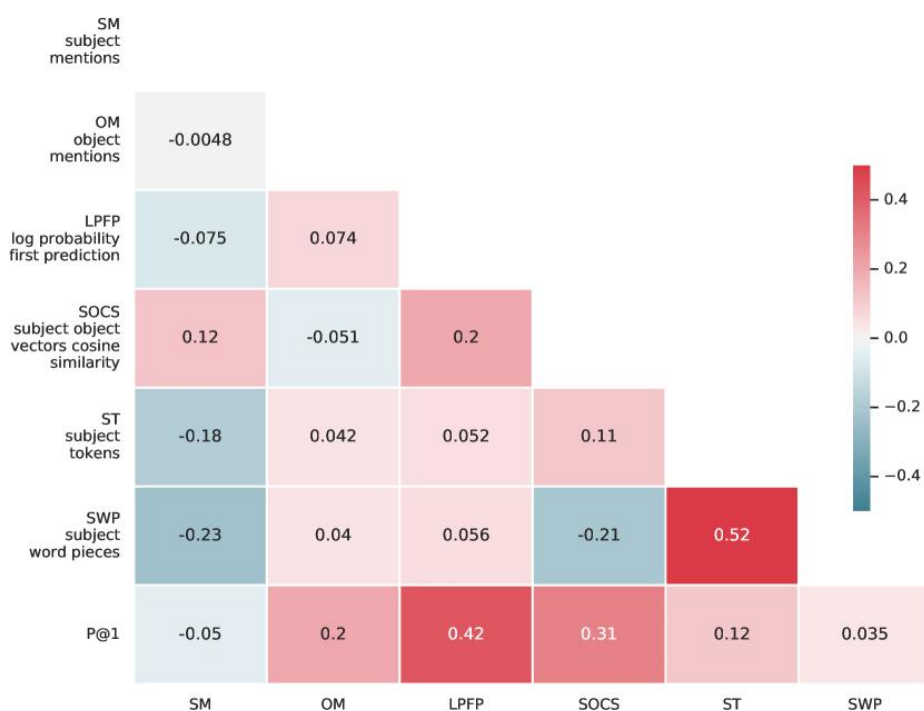


图 3: T-REx 上 BERT-large 模型的 $P@1$ 与一组指标的皮尔逊相关系数: SM 和 OM 分别指 BERT 训练语料 7 中提及主题和对象的次数; LPFP 是与第一次预测相关的对数概率分数; SOCS 是主题和对象向量之间的余弦相似度(我们使用 spaCy 8); ST 和 SWP 分别是主题中采用标准标记化和 BERT WordPiece 标记化的标记数。

我们排除了所有排列不足 10 次的事实。

为了了解预训练语言模型的性能如何随特定事实的不同查询方式而变化,我们分析了每个关系中最多 100 个随机事实,我们从 T-REx 中随机选择了维基百科中的 10 个对齐句子。

在每个句子中,我们都掩盖了事实的对象,并要求模型对其进行预测。对于我

们的几个语言模型来说，这也是在测试它们从训练数据中记忆和回忆句子的能力，因为这些模型都是在维基百科上训练的（见表 1）。

图 4 显示了每个事实 10 次查询的平均排名分布。两个 BERT 模型和 ELMo 5.5B 的变异性最小，而正确对象的平均排名接近前列。令人惊讶的是，ELMo 原始模型的性能与 BERT 相差无几，尽管该模型在训练过程中没有看到维基百科。Fairseq-fconv 和 Transformer-XL 的预测结果差异较大。需要注意的是，与 fairseq-fconv 和 Transformer-XL 相比，BERT 和 ELMo 5.5B 在维基百科中的训练量更大，因此在训练过程中可能看到了更多包含测试查询的句子。

概念网

ConceptNet 语料库的结果与 Google-RE 和 T-REx 中事实知识检索的结果一致。BERT-large 模型的性能一直是最好的，它能够检索出与事实知识水平相近的常识性知识。表 3 的下半部分显示了 BERT-large 在随机抽样例子中的生成情况。语言模型生成的一些概念除了语法正确外，还出奇地合理。

SQuAD

接下来，我们将评估我们的系统在开放域掐词式问题解答上的表现，并与有监督的 DrQA 模型进行比较。表 2 显示了 BERT-large 与 DrQA 开放域 QA 系统在掐词 SQuAD 任务上的性能差距。请再次注意，预训练的语言模型是完全无监督的，没有经过微调，也无法访问专用的信息检索系统。此外，在比较 DrQA 和 BERT-large 的 $P@10$ 时，我们发现两者的差距非常小（BERT-large 为 57.1，DrQA 为 63.5）。

6 讨论和结论

我们对现有公开预训练语言模型中的事实和常识性知识进行了系统分析，发现 BERT-large 能够比其竞争对手更好地召回这些知识，而且与非神经和有监督的其他模型相比，其召回水平具有明显的竞争力。需要注意的是，我们并没有比较相应架构和目标捕捉特定文本知识的能力，而是将重点放在现有预训练模型权重中的知识上，这些模型被用作许多研究人员工作的起点。了解我们常用的模型和学习算法捕

提到了数据的哪些方面是一个至关重要的研究领域，本文是对许多关注数据语言特性的研究的补充。

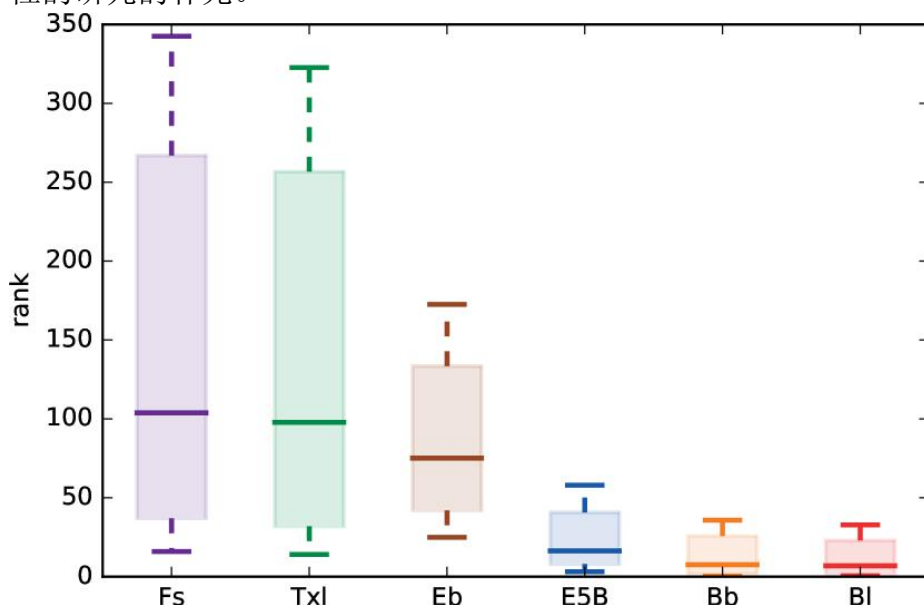


图 4: 在 T-REx 中, 每个关系中 100 个随机事实的 10 种不同提法的平均等级分布。ELMo 5.5B 和 BERT 的两个变体对查询的框架最不敏感, 但也最有可能在训练期间看到查询句。

我们发现, 从文本中提取知识库的难度并不小, 其性能与直接使用预训练的 BERT-large 算法相当。尽管我们只向关系提取基线提供了可能表达目标事实的数据, 从而降低了出现假阴性的可能性, 同时还使用了慷慨的实体链接甲骨文 (entity-linking oracle)。我们怀疑 BERT 可能会因为处理的数据量更大而具有优势, 因此我们将 Wikitext-103 作为额外数据添加到关系提取系统中, 结果发现性能没有显著变化。这表明, 虽然关系提取的性能可能很难随着数据量的增加而提高, 但在不断增长的语料库中训练出来的语言模型在未来可能会成为从文本中提取的传统知识库的可行替代品。

除了使用 LAMA 探针测试未来的预训练语言模型外, 我们还想量化事实知识的回忆与不同自然语言模板之间的差异。此外, 对于我们的评估设置来说, 评估多标记答案仍然是一个有待解决的难题。

燕山大学毕业设计（论文）指导教师意见书

学院：信息科学与工程学院


专业(方向)：软件工程

| | | | | | | |
|---|-------------------|--|----|-----|----|----------|
| 题目名称 | | 基于知识图谱的足球知识问答系统 | | | | |
| 学号 | | 202011200664 | 姓名 | 米亦宸 | 班级 | 软件工程20-3 |
| 序号 | 评审项目 | 指标 | | | 满分 | 评分 |
| 1 | 项目过程的管理意识 5分 | (2分) 1.1 能独立查阅文献和调研；能较好地写出开题报告和中期报告，项目实施过程中各要素组织基本齐全、合理； (3分) 1.2 有综合、收集和正确利用各种信息及获取新知识的能力，能考虑非技术指标因素。 | | | 5 | 4 |
| 2 | 自我学习意识和工作态度 5分 | (2分) 2.1 参加了毕业设计（论文）的各个训练环节，按期独立完成规定的工作任务，可以很好的综合运用所学专业知解决遇到的问题；能够独立查阅相关资料内容，可以很好的解决实际问题。 (3分) 2.2 遵守纪律，学习认真；尊敬指导教师，能够听取指导教师的意见和指导；不存在学术不端行为。 | | | 5 | 5 |
| 是否同意参加答辩： <input checked="" type="checkbox"/> 同意 <input type="checkbox"/> 不同意 | | | | | 总分 | 9 |
| 评语： 该生态度端正，学习认真，进取心强。 论文符合撰写规范，同意答辩！ <div style="text-align: right;"> 指导教师签名：王晓妍 日期： </div> | | | | | | |

燕山大学毕业设计（论文）评审表（评阅人用）

学院：信息科学与工程学院（软件学院）

专业（方向）：软件工程

| | | | | | | |
|--|-------------------|--|-----|-----|------------|----|
| 题目名称 | | 基于知识图谱的足球知识问答系统 | | | | |
| 学号 | 202011200064 | 姓名 | 米亦宸 | 班 级 | 20 级软件 3 班 | |
| 序号 | 评审项目 | 指标 | | | 满分 | 评分 |
| 1 | 报告和论文的撰写质量 5 分 | <p>（3 分）1.1 逻辑结构。毕业设计（论文）主题基本明确、结构基本合理、有一定层次性，引用资料基本准确，附录材料与毕业设计（论文）基本相符。</p> <p>（2 分）1.2 写作能力。毕业设计（论文）语言基本流畅，整体内容能够体现设计主题；文字表述、语法应用、书写格式、图表注释、缩略词等基本符合国家相关标准；参考文献标注基本规范、来源清楚。</p> | | | 5 | 4 |
| 2 | 外文资料阅读与翻译 5 分 | <p>（2 分）2.1 阅读能力。外文原文与毕业题目相关，来源于计算机领域期刊或者会议、不少于 6 页。</p> <p>（3 分）2.2 翻译能力。中文翻译与外文原文对应，翻译准确、流畅。</p> | | | 5 | 4 |
| | | | | | 总分 | 8 |
| <p>评语：</p> <p style="text-align: right;">评阅人签名：  日 期：</p> | | | | | | |

燕山大学毕业设计（论文）答辩评审表

学院：信息科学与工程学院（软件学院）

专业（方向）：软件工程

| 题目名称 | | 基于知识图谱的足球知识问答系统 | | | | |
|--|-------------------------|---|-----|-----|------------|----|
| 学号 | 202011200064 | 姓名 | 米亦宸 | 班 级 | 20 级软件 3 班 | |
| 指导教师姓名 | | 王晓妍 | 职称 | 副教授 | | |
| 序号 | 评审项目 | 指标 | | | 满分 | 评分 |
| 1 | 功能实现的完备度和性能达标情况 10 分 | （5 分）1.1 设计能力。可以很好的应用开发工具设计符合国家或行业标准的毕业设计（论文），体现一定分析解决本专业实际问题的综合设计能力，基本达到选题的目标要求。 （5 分）1.2 系统能力。体现出较为扎实的专业基本知识，基本能够综合运用开发工具解决毕业设计（论文）相关问题。软件系统能够正常运行，数据正确，代码命名规范、缩进风格统一，能够准确的无误的讲解代码的运行逻辑。 | | | 10 | 7 |
| 2 | 软件测试能力 5 分 | (5 分) 2. 软件测试能力。软件测试过程完备，测试方案合理。 | | | 5 | 3 |
| 3 | 创新和发展意识 5 分 | （2 分）3.1 创新意识。设计能够体现一定的创新思路。 （3 分）3.2 发展意识。设计方案正确，设计方法得当，体现一定分析解决本专业实际问题的综合设计能力。 | | | 5 | 3 |
| 4 | 工作总结和综合表达能力 10 分 | （5 分）4.1 工作总结。答辩表述清楚，语言基本流畅，整体内容能够体现毕业设计（论文）主题；体现出较为扎实的专业基本知识，核心概念基本明确。 （5 分）4.2 成果展示。态度认真，对完成的毕业设计（论文）内容完全掌握；回答问题有理有据，基本概念清楚，主要问题回答准确，有一定深度。 | | | 10 | 7 |
| | | | | | 总分 | 20 |
| <p>评语：</p> <p>答辩委员会小组成员：</p> <p>姓名职称（签名）：徐玉强 副教授 姓名职称（签名）：米亦宸 教授</p> <p>姓名职称（签名）：李书华 副教授 姓名职称（签名）：_____</p> <p>姓名职称（签名）：_____ 姓名职称（签名）：_____</p> <p>答辩委员会（小组）负责人签名：米亦宸</p> <p>年 月 日</p> | | | | | | |

燕山大学毕业论文开题和中期考核评分表

姓名：米亦宸 学号：202011200064 班级：20 级软件 3 班

论文题目：基于知识图谱的足球知识问答系统

开题考核：

| 开题考核(满分 20 分) | | | |
|-------------------------|--------------------|----------------------------|------|
| 社会发展背景的认识 和选题意义(5 分) | 国内外研究现状 分析(5 分) | 具备软件需求分析和技 术选型的能力(10 分) | 开题成绩 |
| 4 | 4 | 8 | 16 |

中期考核：

| 中期考核(30 分) | | | |
|---------------------|---------------------|--------------------|------|
| 软件设计方案的质量 (15 分) | 问题研究能力的考 核(10 分) | 项目管理能力的考 核(5 分) | 中期成绩 |
| 12 | 8 | 4 | 24 |

燕山大学毕业论文评审和答辩评分表

指导教师评分：

| 导师评分(满分 10 分) | | |
|--------------------|----------------------|--------|
| 项目过程的管理意识 (5 分) | 自我学习意识和工作态度 (5 分) | 导师评分汇总 |
| 4 | 5 | 9 |

指导教师签字：

王健妍

年 月 日

毕业(设计)论文答辩：

| 答辩考核(满分 40 分) | | | | | | |
|-----------------------------------|---------------------|--------------------------|------------------------------|------------------------|------------------------|----------|
| 功能实现的 完备度和性 能达标情况 (10 分) | 软件测试 能力 (5 分) | 创新和 发展意 识 (5 分) | 工作总结 和综合表 达能力(10 分) | 报告和论 文撰写质 量(5 分) | 外文资料 阅读与翻 译(5 分) | 答辩成 绩 |
| 7 | 3 | 3 | 7 | 4 | 4 | 28 |

答辩组组长签字：

田利军

年 月 日

综合其开题考核、中期考核、导师评分、评阅评分、答辩成绩，

该本科生毕业(设计)论文的总成绩为：73 + 14 = 87。

(□优秀、□良好、☒中等、□及格、□不及格)

郭陈东