基于大模型的旅游咨询智能体设计与实现

王幸欣，兰红

（江西理工大学，信息工程学院，江西 赣州 341000）

摘　要：随着旅游业发展与网络信息过载，传统旅游规划及大语言模型在旅游服务上存在局限。为此，构建融合 RAG 技术的智能旅游 Agent 意义重大。本文设计并实现了基于大语言模型的旅游 Agent 系统，采用前后端分离架构，前端基于 Vue3 与 Vite，后端基于 FastAPI，使用 LangChain 构建 Agent，Chroma 向量数据库构建 RAG。客户端支持对话式查询，提供旅游路线规划等服务；管理端可上传 PDF 构建本地知识库。该系统有效提升模型问答准确性，为用户提供便捷智能的旅游助手，满足了旅游出行的个性化需求。

关键字：大语言模型；Agent；RAG；LangChain

Abstract：With the development of the tourism industry and the overload of online information, traditional tourism planning and large language models have limitations in providing tourism services. Therefore, it is of great significance to build an intelligent tourism agent that integrates RAG technology. This paper designs and implements a tourism agent system based on large language models, adopting a front-end and back-end separation architecture. The front-end is based on Vue3 and Vite, while the back-end is based on FastAPI. LangChain is used to build the agent, and Chroma vector database is used to construct RAG. The client supports conversational queries and provides services such as tourism route planning. The management end can upload PDFs to build a local knowledge base. This system effectively improves the accuracy of model answers, providing users with a convenient and intelligent travel assistant that meets the personalized needs of travel.

Key words：LLMs；Agent；RAG；LangChain

0 引言

随着旅游业的蓬勃发展和互联网旅游内容的海量增长 [0]，旅游领域陷入信息过载困境，如何助力游客快速制定个性化游览路线成为研究焦点。实际出行时，旅游计划受出行对象、家庭结构、游览兴趣、预算水平等复杂因素制约，而传统旅游路线规划方法，无论是依赖精确数学建模，还是基于用户生成内容，都存在个性化不足、实时性差的弊端。即便主流旅游平台提供丰富攻略和行程推荐，用户仍需大量主观判断与重复操作，难以满足对高效、个性化服务的需求。

以 Deepseek、ChatGPT[2]、Qwen 为代表的大语言模型在自然语言理解与生成方面表现极佳，可理解复杂指令、流畅对话并具备一定推理能力。但 LLM 自身无法实时感知世界、调用外部数据，在旅游行业中，若景点道路维修、店铺营业时间变动等，通用 LLM 无法获取最新消息，影响出行规划。为此，RAG 和基于 LLM 的 Agent 机制应运而生。RAG 能从外部知识库检索信息补充给 LLM，增强回答准确性与可溯源性；Agent 则具备行动能力，可调用工具执行任务，动态响应世界变化，在旅游场景中能结合实时信息生成个性化建议。

然而，现有的旅游助手类应用仍存在诸多缺陷。功能方面，天气、路线、推荐信息分散于不同平台，缺乏统一入口。在交互能力上，多数助手仅支持关键词搜索，难以理解用户自然语言需求。并且存在实时性不足，数据更新滞后的问题，无法提供及时反馈与

应急建议，大多数应用个性化欠缺，并未根据用户进行个性化推荐。

鉴于此，本文融合 RAG 检索增强技术，构建可调用外部服务的智能旅游 Agent，赋予 Agent 获取时间和地图信息等工具能力，结合用户个性化需求与实时信息，实现动态、个性化的旅游路线规划，提升用户旅游体验，并搭建了相应的 Web 应用，用户在网页前端输入问题即可与 Agent 进行交互，为旅游与 AI 行业发展提供新方向。

1 RAG 模块的设计与实现

1.1 RAG 工作流程介绍

大语言模型本质上通过在海量文本上学习下一个词的概率来生成内容的，并不具备理解世界知识的能力,有研究者提到 LLMs 在学习过程中高度依赖词语的共现关系和位置接近性来构建事实知识，这种依赖可能导致模型忽视词语之间的语义关联，生成的内容语法通顺，看似合理但是存在与事实相悖或者是逻辑错误的部分，这被称为“幻觉现象”[3]。此外，大模型的上下文窗口长度有限，这导致它在处理长文本或多轮对话时，早期信息可能被遗忘，从而引发上下文不一致问题。而由于生成内容是从概率分布中采样而来，并非根据已有文档进行生成，生成的结果通常缺乏明确出处，难以验证其正确性[4]。解决这一现象的措施有很多，例如有知识图谱，提示工程与 CoT[0]，大语言模型微调以及 RAG 检索增强等技术。RAG 通过为 LLMs 提供可信的参考文档，从而降低 LLMs 产生“幻觉”的概率[6]，同时还能通过引入大量垂直领域或者是私人领域的文档，RAG 技术的整体流程如图 1 所示，分为离线数据处理和在线检索两个过程，经过离线处理，数据会以特定的格式和排列方式存储在知识库中，在线检索就是利用大模型针对用户的 query 查询对构建知识库查询的过程。下面以基于语义检索的RAG对其构建过程进行讲解。

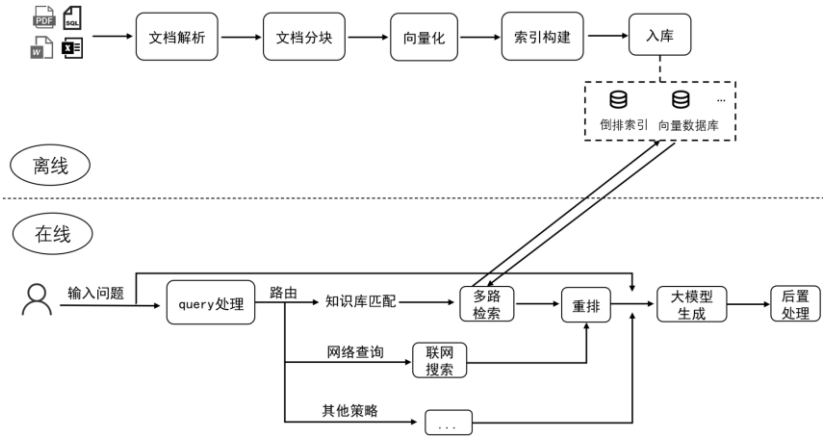

图 1 RAG 的整体流程图

离线数据处理中，首先是进行文档解析，将各种类型的数据转换成文字并进行清洗和格式统一。在这之后可以选择对文本进行清洗，例如修正 OCR 错误。接着对解析好的文档进行分块，也叫做文本切分。接着使用嵌入模型将每个文本块映射到高维向量空间，并建立索引，生成的向量连同对应的文本 ID 和元数据一起存入专门的向量数据库。常用的向量数据库包括 Milvus、Weaviate、Chroma 等,这些数据库针对高维向量进行了优化，可以快速执行近邻检索。同时，还能使用传统搜索引擎为原始文本建立反向索引，以支持基于关键词的检索。目的都是让相似语义的查询和文档向量在空间中距离较近，以便后续检索时快速找到相关内容。通过这一系列的操作，离线数据处理完毕后，完成了对特定的文档构建专属的知识库以及检索方式。

随后只需等待用户 query 的到来，即在线检索。如图 2 所示。将 query 输入大模型，首先对 query 进行处理，通常包括 query 解析以及向量化处理，并使用 Embedding 模型

对 query 进行向量化。LLM 设计处理问题的步骤并判断是否需要检索外部知识以及采取何种检索方式，这一步骤通常被称为“路由”。如果需要检索的话，将利用向量化后的 query 与向量数据库中的文档进行相似度计算，最终将检索到的相关上下文与用户问题一起拼接返回给大模型生成答案。这样一来，既利用了静态知识库，又为 LLM 提供了更加实时且精准的背景信息，有效降低了模型幻觉，提高了答案准确率。

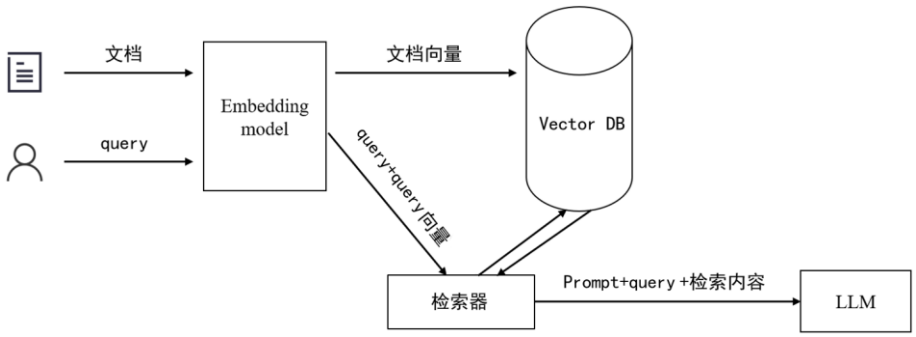

图 2 RAG 系统中的检索流程图

1.2 RAG 设计框架

本文的 RAG 设计分成两个模块，一个是 PDF 处理模块，另一个是向量数据库连接模块。RAG 的设计框架如图 3 所示。PDF 处理模块的其中一个步骤使用到了向量数据库模块。PDF 处理模块先对 PDF 加载并提取文本内容，由于 PDF 中会存在一些图表信息，所以在设计内容提取的时候考虑了图表信息，将其作为特殊文档添加到检索库中。接着将文本切分成小块，支持中文分隔符。为了避免重复的内容影响检索的质量，设计了去重模块，对分块后的文本进行去重优化。最后将处理后的文档分批插入 ChromaDB，此处就涉及到了向量数据库模块，在文档插入的过程中设计了进度显示条。在向量数据库模块中，主要分成了 ChromaDB 连接、文档插入以及检索器配置三个功能。

RAG 设计的主要优点总结下来有如下四个点：

（1）图表提取：本项目是利用 PDF 构建知识库，不仅提取文本，还对 PDF 中存在图表信息也进行了提取，增强了检索的丰富性。

（2）分块策略：使用递归分块器并支持中文分隔符，提高了分块的质量。

（3）去重优化：避免了重复内容影响检索结果。

（4）批量处理：分批处理 PDF 文件和入库，可以提高性能和稳定性。

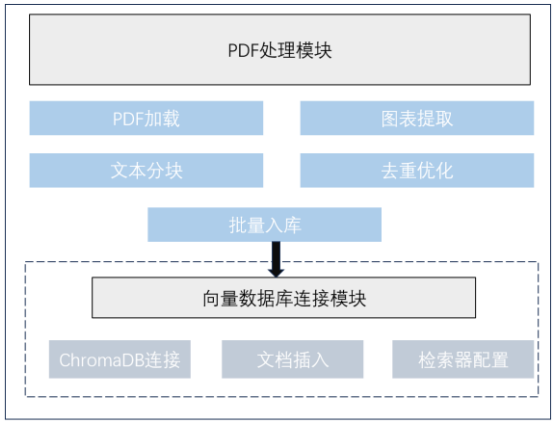

图 3 RAG 设计框架图

整个 RAG 的工作流程如图 4 所示，首先进入初始化模块，主要是对 PDF 处理模块以及 ChromaDB 连接模块进行初始化。PDF 处理模块中需要设置 PDF 文件所在目录、Embedding 模型、文本分块大小等参数，在检索环节涉及到了相似度阈值以及检索数量，同样需要对其进行初始化。向量数据库连接模块中设置 ChromaDB 持久化路径等参数，另外还需要对检索器参数进行初始化。对所需要的参数初始化完毕后，根据 query 对文档进行检索，并对检索到的结果进行去重和排序，构建 RAG 查询链对检索到的内容进行整理，利用大语言模型生成最终的回复。

图 4 系统 RAG 的工作流程设计图

1.3 数据预处理模块

RAG 的知识库基于 PDF 文档检索，因此数据预处理模块针对 PDF 文档设计。PDF 文档处理模块将 PDF 文档处理后存入 ChromaDB 向量数据库，封装了加载、提取、分块、去重和入库等操作。

实现批量处理，筛选出以.pdf 结尾的文件，生成待处理列表。接着从单个 PDF 文档中提取文本和图表信息，分析页面布局，定位图表元素并提取位置和描述信息，将图表信息封装成字典作为特殊文档处理。这样，RAG 系统能处理多模态信息，增强对复杂文档的理解能力。

提取文本后进行分块处理，分块处理能够提高检索效率和准确性。根据预定义字符序列递归分割文本，设置分块大小为 4000，分块重叠大小为 1000，确保上下文信息连续。分块时为每个分块添加来源和页码信息，方便后续追踪引用。文本切分的分隔符包含中英文标点和换行符，优先级从高到低为段落分行符、句子标点符号、空格，保证文本在语义完整处切分。最后，使用集合去重，避免重复文本块存入向量数据库，减少存储空间浪费，提高检索效率。

1.4 向量数据库构建

文本处理完成后，将文档向量插入 Chroma 向量数据库。向量数据库是 RAG 系统的核心，Chroma 向量数据库存储文档的向量表示，支持高效相似性检索。选择 text-embedding-v3 作为嵌入生成模型，它对比 text-embedding-v2 扩展了意大利语、波兰语等五十多种语种，编码输入长度扩展至 8192，具有高性能低成本的特点。

1.5 向量数据库连接模块

向量数据库是 RAG 系统存储和检索知识的关键基础设施。向量数据库连接模块抽象了与 Chroma 向量数据库的交互，包括 ChromaDB 连接、文档插入和检索器配置功能，为 RAG 的 PDF 文档处理模块提供统一接口。

ChromaDB 采用本地连接，构造一个检索器初始化接口，可设置检索类型、数量、相似度阈值和过滤器，在构建 RAG 问答链时具体赋值。

1.6 RAG 问答链构造

向量数据库构建完成后，即可构造 RAG 问答链。RAG 问答链是基于 RAG 构造的问答链，作为 Agent 的工具模块之一进行使用。

RAG 问答链包含文档检索器、提示词模板和大语言模型三个核心组件。本系统采用语义检索，检索器配置基于相似度分数的阈值过滤机制，相似度阈值为 0.3，检索数量为 20，只有相似度超过 0.3 且排序在前 20 的文档才会进入候选集传给 LLM 生成回答。

提示词模板明确了模型的角色，强调基于检索内容回答，不知道答案时保持诚实，尽量简洁回答。大语言模型选取 Qwen-max，它是通义千问 2.5 系列千亿级别超大规模语言模型，支持中文、英文等多语言输入，适合问答场景。

2 Agent 的设计与实现

2.1 Agent 详细设计

本系统的 Agent 基于 LangChain[7]框架搭建，目标是与用户之间通过自然语言进行交流，可以记住用户的历史消息支持多轮对话，通过各个组件之间的协同，能够准确理解用户意图，通过调用合适的工具或知识库给用户生成具备个性化的旅游攻略，Agent 的设计是本系统的关键所在。

Agent 的核心认知能力取决于大语言模型，和 RAG 问答链所选取的模型相同，Agent 也是选择了 Qwen-max 大语言模型作为驱动。

Agent 调用的外部工具赋予了智能体行动的能力，tool-calling 技术是指在 LLM 自动识别并调用外部工具（如 API、数据库等）来完成其无法独立完成的任务的一种能力，这项技术决定了智能体的上限。因为模型并不是什么都知道，它只是擅长理解和生成语

言。在现实中，很多问题光靠理解和生成语言是无法解决的，比如询问 LLM 今天赣州的天气怎么样，LLM 并不能给出正确的回答，当赋予了 LLM 查询天气的 API 后，LLM 就会生成调用查询天气 API 的参数，让外部工具返回参数化结果，大模型根据返回的结果来生成最终的回复，具体的 Tool-calling 运行流程如图 5 所示。

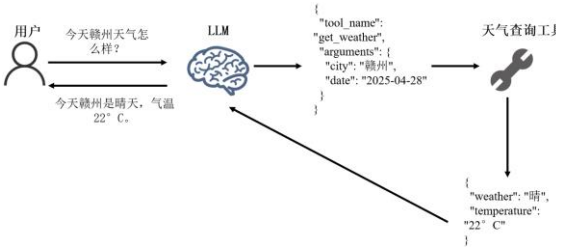

图 5 Tool-calling 运行流程图

为扩展 Agent 与现实世界交互的能力，Agent 集成了一系列外部工具，有地理位置编码工具、骑行路线规划工具、驾车路线规划工具、附近位置搜索工具、RAG 工具、联网搜索工具、天气查询工具、获取时间工具以及时效性检测工具，其中地理位置编码负责利用地名为骑行路线规划工具以及驾车路线规划工具提供地点的准确经纬度。

时间工具通过 python 内置库实现，联网搜索工具基于DuckDuckGo搜索引擎构建，Agent 会分析用户的查询来使用搜索引擎搜索相关的网页内容。

调用高德 API 构建了地理编码工具、驾车路线规划工具、骑行路线规划工具以及附近位置搜索工具，驾车路线规划工具以及骑车路线规划工具能够得到路线的具体信息以及行驶时间，Agent 能够根据用户的特定需求合理利用路线规划工具，例如用户需要从 A 地到 B 地点，图中需要经过指定的一系列地点的时候，Agent 能够利用路线规划工具为其推荐合理的路线。地理编码工具负责将地理位置转换成经纬度信息，方便给路线规划工具以及附近位置公司工具提供准确的经纬度信息，而不是依靠大语言模型内部的知识生成地点的经纬度信息，这样生成的经纬度信息往往既不准确也不稳定。

调用和风天气API构建天气详情获取工具，能够获得气压、湿度、温度、紫外线强度、天气状况等等天气信息，Agent 拥有了天气查询工具后，便能够从天气角度为用户生成穿搭、活动以及出行建议，如果不结合天气信息生成出行建议，当用户询问“我明天想去厦门玩，可以去哪些地方呀？”，然而现实情况是明天会下暴雨，那么 Agent 有可能推荐用户在暴雨天去海边旅游，这是极其不合理的，考虑了天气状况后，将能够避免这样的回复生成。

此外，RAG 功能也是作为 Agent 的工具进行使用，是基于先前已经构建的 RAG 问答链进行构建。由于旅游问题普遍具有时效性，所以设计了一个时效检测工具，用于判断用户当前的问题是否具有时效性，如果具有时效性的话，Agent 将会调用联网搜索功能。这些工具使 Agent 能够获取实时天气、地理位置信息、联网搜索信息，并查询本地知识库，极大地增强了其实用性。当 LLM 根据提示词和用户输入判断需要使用某个工具时，会生成相应的工具调用请求，AgentExecutor 则实际执行该工具并将结果返回给 LLM，供其生成最终回复。

2.2 Prompt 提示词工程设计

本系统基于CO-STAR框架进行改进设计 Agent 的提示词，CO-STAR 框架从 Context、Objective、Style、Tone、Audience 以及 Response 六个维度进行设计 Prompt。Context 模块明确 Agent 的身份为“小旅宝”，是一个热情、知识渊博的智能旅游助手。Objective 模块指导 Agent 提供准确、有趣、实用的旅游信息和攻略，强调基于检索内容回答，禁止编造。Steps 模块定义了 Agent 处理请求的标准化流程，如先进行时效性判断，再根据结果选择知识库检索或联网搜索，并结合其他工具。Tools 模块列出了 Agent 可用的工具以及每个工具的作用，Arguments 模块强调了工具调用的逻辑，对工具的使用进行了补充说明。Response Style 模块定义了 Agent 回答用户时的语气以及风格，要求回答风趣亲和，使用网络热词并搭配相关表情，使得生成的内容更加有趣。在构建 Prompt 的时候通过对话历史占位符向 LLM 提供了先前的对话内容，帮助 Agent 理解当前对话的上下文。同时通过

Agent 中间步骤占位符传入了 Agent 在思考和调用工具过程中的中间步骤和结果，供 LLM 参考，方便进行后续的相关决策。

3 系统详细设计

3.1 系统总体功能设计

基于 Web 网页搭建可交互的旅游 Agent 智能助手，基于 LangChain 框架开发 Agent 功能，使用前后端技术开发功能接口并进行可视化。图 6 是系统的功能图，系统的功能模块分成普通用户以及管理员两大模块进行设计。普通用户具有会话管理、天气展示、快捷提问以及智能体使用等四个功能模块，在智能体模块中，具备检索增强生成、获取时间、地图搜索、天气查询以及联网搜索等五个功能模块，管理员则具有登录、文档上传、知识库更新以及退出登录这四个功能模块。

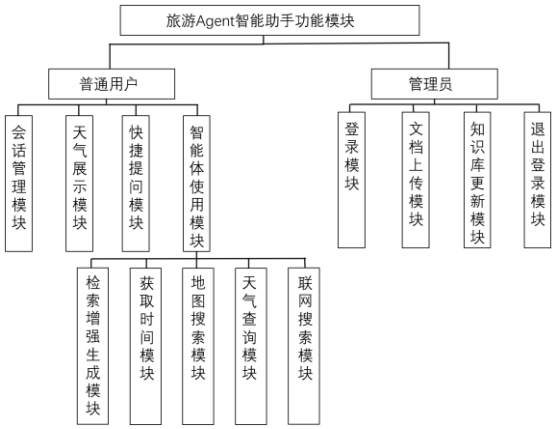

图 6 系统功能设计图

3.2 系统架构设计

系统开发采用前后端分离架构的客户端-服务器架构，前端依赖浏览器原生的 Fetch API 进行网络请求，前后端之间通过 Json 进行数据交流，采用 RESTful API 设计规范，确保前后端交互的一致性和可维护性，支持异步请求处理，实现流式响应机制，使大模型的生成内容能够实时展示，支持复杂的旅游信息查询和智能对话功能。

3.3 前端架构分析

前端项目基于 Vue3 框架构建，整体架构如图 7 所示。最上层为视图层，负责页面的整体结构和视觉呈现，由 Vue 组件构成，结合 Html 和 Css 来编写页面，并展示各类静态资源。路由层通过 Vue Router 实现页面导航与切换，并利用路由守卫机制对管理员页面进行权限控制，提升了系统的安全性和用户体验。数据层分为缓存和持久层，缓存主要依托 Vue 的响应式状态管理，实现页面数据的实时响应和组件间的数据共享。持久层利用浏览器的 localStorage 和 sessionStorage 等本地存储机制，实现了会话信息、历史记录和安全验证 token 等关键数据的本地保存，确保用户在页面刷新或关闭后仍能保留相关数据。最底层为通信层，负责前端与后端的具体数据传输。系统不仅采用原生 Fetch API 实现异步请求，还集成了 Axios 库以便于统一管理 HTTP 请求和错误处理，并通过 Vite 的代理配置解决开发环境下的跨域问题。此外，通信层还支持流式响应处理，适应大模型推理场景下的数据实时传输需求。

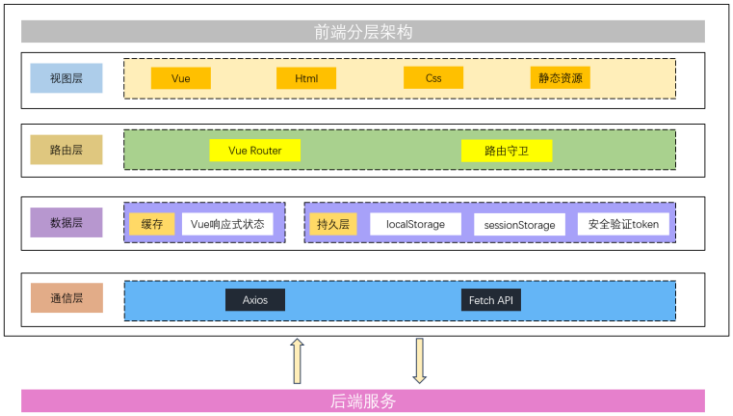

图 7 系统前端架构图

3.4 后端架构分析

后端系统采用了分层架构设计，整体结构如图 8 所示。自下而上，系统以数据源为基础，数据源分为结构化数据和非结构化数据两大类。结构化数据包括 MySQL 数据库中的管理员信息、服务器存储的城市列表以及 JSON 格式的旅游景点等基础图片链接等数据，这些数据主要用于天气查询、用户管理和网页展示。非结构化数据则以 PDF 文档等旅游攻略资料为主，这些文档经过分块、文本提取和向量化处理等操作后，为后续的语义检索提供原始素材。

在数据层，用于网页天气展示的结构化数据通过爬虫模块采集。非结构化数据主要通过向量化处理模块，将 PDF 文档转化为可用于语义检索的向量表示。

所有处理后的数据最终流向数据存储层，系统采用 ChromaDB 向量数据库存储经过向量化处理的文本数据，同时也保留了部分本地数据文件和MySQL数据库用于静态信息和结构化数据的快速访问。

业务逻辑层是整个后端的核心部分，包括会话管理、RAG、问答链、Agent 以及第三方 API 集成等模块。会话管理模块负责维护用户的多轮对话历史，实现上下文记忆。RAG 模块结合了知识库检索与大语言模型生成，提升了问答的准确性和丰富性。问答链模块主要实现 RAG 链，将用户输入、知识检索和答案生成串联起来得到最终回复，目的地推理链则基于LLM实现自动从用户输入中提取目的地名称。Agent 模块基于 LangChain 实现，具备多工具协同、动态决策和多轮对话能力，统一调度所有第三方 API 的调用，如天气查询、地图检索和网络搜索等，极大地扩展了系统的外部能力。

最上层为数据接口层，系统通过 FastAPI 框架对外提供 RESTful 接口，支持包括天气查询、景点推荐、对话问答等多种服务。CORS 中间件保证了前后端的安全通信，流式响应机制实现了实时数据传输。

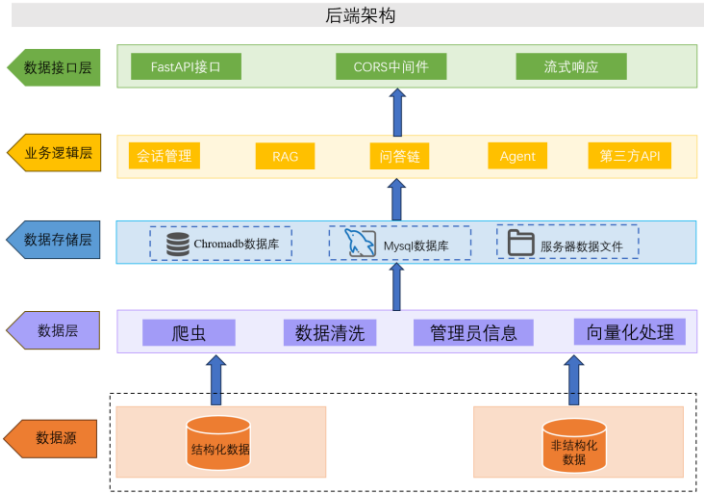

图 8 系统后端架构图

4 系统实现效果

4.1 系统前后端实现效果

Agent 搭建完毕后，需要提供可视化交互系统，系统分为两个页面，分别是主页面和管理员页面。主页面无需验证即可进入，任何人都可以使用，管理员页面提供给管理员使用，需要登录管理员账号才能进入。

在主页面，分成了三个部分，左侧是会话管理栏，中间是对话栏目，右侧是天气展示栏，如图 9 所示。

对于会话管理栏，可以点击箭头进行收起和展开，点击加号能够创建新的会话。对于每个会话框可以点击右侧三个点对会话进行命名，每个会话也具备删除的功能。对于会话管理栏目中的每个会话，当会话已经命名过，那么前端将会调用后端所搭建的地名推测链来获得会话中所包含的地名，并调用天气接口在右侧天气栏目展示会话所包

含地名的天气信息。如果检测不到，那么就会根据会话所包含的聊天历史信息，获得会话中用户最近咨询的地点的天气信息进行展示，如以上都未能检测到地名，那么会默认显示首都北京的天气信息。

在对话栏目，用户可以在输入框输入问题后点击上传按钮，此时前端将会向后端发送调用 Agent 的请求，等待 Agent 生成回复后，前后端采用流式输出实时获取 Agent 的回复，边获取边生成。在输入框下方是随机生成的三个常见的旅游问题，用户可以直接点击就能对 Agent 进行咨询。

右侧天气栏目配合景点信息以及温度和天气状况对目的地的天气进行展示。在右上角有管理员登录按钮，点击即可进入登录框进行登录。

图 9 系统主页面

管理员登录框如图 10 所示，需要输入用户名和密码才能进行登录。确保后台只有管理员才能进入，实现了管理员认证授权机制。该机制以当前广泛应用的 JWT 技术为核心，提供了一种无状态且安全的身份验证方案，保障仅经过授权的管理员用户能够访问和操作关键的后台管理模块。用户通过登录入口触发登录框。在登录框中，管理员输入用户名和密码进行提交，前端将这些数据封装并通过 POST 请求发送至后端的登录验证接口。后端接口在收到凭证后，会查询 MySQL 数据库以验证用户身份及密码的正确性。

图 10 管理员登录页面

若后端成功校验凭证，便会签发一个 JWT，并将其作为响应数据回传给前端。前端在接收到此 JWT 后，会将其安全地存储于浏览器的 localStorage 中，此 Token 可作为该管理员后续操作的有效身份凭据。在后续需要访问上传 PDF 文件以及创建知识库等需要授权的后端 API 接口时，前端会在 HTTP 请求的 Authorization 头部以 Bearer <token>的标准格式附带此存储的 JWT。此外，为防止未经认证的用户通过直接操作浏览器地址栏访问受保护的管理页面，前端路由还配置了导航守卫。这些守卫在路由跳转前检查目标路由是否需要认证权限以及 localStorage 中是否存在有效的 JWT，若不满足条件，就会将用户重定向至主页面。管理员的登出操作会清除本地存储的 JWT，并引导用户返回主页面。

管理员界面如图 11 所示，管理员可以点击上传 PDF 文件以及创建 RAG 知识库按钮，调用后端接口实现相应的功能，页面右上角可以点击退出登录按钮返回主页面。

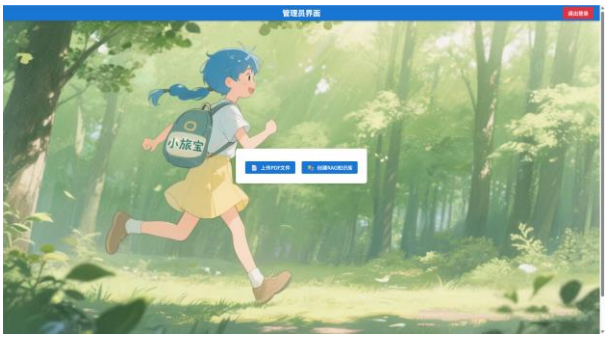

图 11 管理员页面

4.2 Agent 实现效果

4.2.1 对比 Qwen-max 大语言模型的输出

为了验证 Prompt 对 Agent 的语气风格

设置已经生效，对本系统的 Agent 进行提问与 Qwen-max 大语言模型的输出进行对比。输入“你是谁？”，图 12 展示了本文所搭建的 Agent 与 Qwen-max 官网大模型所生成的回复对比，说明 Prompt 对 Agent 的角色设定以及回复语言风格的设定已经生效。

图 12 本系统的 Agent 与 Qwen-max 的输出对比

4.2.2 联网功能

当询问 Agent 关于演唱会的消息时，Agent 能够准确判断这是一个需要进行联网搜索以获得实时信息的提问，图 6-13 展示了 Agent 对于演唱会相关活动咨询的回复。其中，Agent 还返回了相关的网页链接，用蓝色字体进行了标注，用户点击即可查阅相关信息。

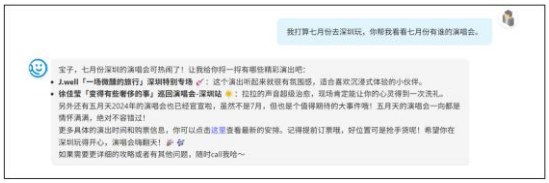

图 13 联网功能效果图

4.2.3 路线规划功能测试

路线规划功能包括驾车路线规划以及骑行路线规划功能，当用户询问从江西理工大学三江校区出发到上犹县阳明湖景区的驾车规划路线时，Agent 能够生成与高德地图所推荐的相同路线。当用户继续追问“如果是骑行呢？”，Agent 也能够给出正确的回复。如图 14 所示，不仅说明 Agent 的路线规划功能没有问题，同时也说明了 Agent 多轮对话功能也成功实现了。由于路线规划功能是基于地理编码工具实现的，如果地理编码不能提供精准的地点经纬度信息，那么也就无法正确完成路线规划的任务。此时，地理编码功能也得到了验证。

图 14 路线规划功能效果图

4.2.4 附近位置搜索功能

当用户的问题需要借助地图工具进行附近地点搜索的时，Agent 进行思考并返回准确的回复。如图 15 所示，当用户询问“江西理工大学三江校区附近有什么奶茶店？”，Agent 对用户意图进行了准确的判断，并输出了准确的回复。

图 15 附近搜索功能样例 1 效果图

当输入“江西理工大学三江校区附近有什么推荐的自然风景？”时，Agent 能够返回公园相关的地点推荐，如图 16 所示。说明 Agent 不仅仅是靠关键词进行搜索，能够理解公园也是属于自然风景类别。

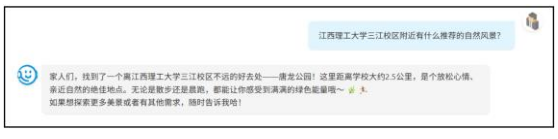

图 16 附近搜索功能样例 2 效果图

4.2.5 时间获取功能

向 Agent 询问“赣州的夏季一般从几月份开始，持续到几月份？”时，Agent 不仅生成了正确的回答，而且还得到了用户发起请求的具体时间，提醒用户现在赣州所处的季节，并给用户生成了穿搭等注意事项。对

话如图 17 所示。

图 17 时间获取功能测试效果图

4.2.6 RAG 检索增强功能

图 18 展示了用来构建知识库的相关 PDF 文档中关于厦门环岛路的介绍，图 19 比较了本文所构建的 RAG 问答链与所调用的 Qwen-max 大模型回答的区别，用户输入的问题是“请你介绍厦门的环岛路”，可以看到本文所构建的 RAG 问答链不仅包含了 Qwen-max 所具备的信息，还检索到了文档中所包含的内容，PDF 文档中所提到的环岛路的别称“黄金海岸线”，在构造的 RAG 问答链中有包含此内容，而在阿里官网对 Qwen-max 进行提问的时候，其输出并不包含 PDF 文档中的内容，说明 RAG 功能构造成功，实现了基于 PDF 文档为旅游 Agent 助手构造知识库的功能。

图 18 RAG 测试样例相关的 PDF 文档内容

图 19 RAG 问答链与 Qwen-max 大模型回答的对比展示图

4.2.7 综合功能

最后验证 Agent 在更复杂场景下需要调用多个工具并需要进行多轮对话时的反应效果，图 20 的对话验证了 Agent 的多轮对话效果，当用户第二次提问的时候会根据用户的历史消息生成回复，并且能够获取到准确的时间信息。Agent 为用户生成的旅游攻略中，考虑了住宿地点、游玩时间、天气、饮食口味以及游玩兴趣等因素。例如，推荐了用户住宿附近的一家具体的饭店享用早餐，在推荐的景点中几乎都是满足用户游玩兴趣的地点，并且结合了天气为用户给出了穿搭建议以及出行注意事项。

图 20 Agent 综合功能样例 1 测试图

紧接上文的提问，用户继续输入问题“第二天我需要去厦门大学一趟办点事情，十三点到达厦门大学，十五点离开。这个事情比较紧急，请你以这个事情为重点，剩下的时间为我安排合理的行程计划，不要让我比较赶，以至于迟到哈。”，如图 21 所示。可以看见 Agent 结合了历史消息所提供的背景为用户的最新提问给出了回复，用户的住宿地点在前文中已经提到过，在后续的答复中 Agent 考虑到了这点，并进行了路径规划为用户考虑了出行时间方面的因素，在出行计划较为特殊的情况下也能生成满意的回复。

图 21 Agent 综合功能样例 2 测试图

5 结语

本文成功设计并实现了一个基于 LangChain 框架的智能旅游助手 Agent，并基于前后端分离架构搭建了 Agent 使用系统，系统所构建的旅游 Agent 智能助手融合了多个外部工具，以 Qwen-max 大语言模型为基础，扩展了其能力边界，能够使用工具为用户提供更多的服务，从而生成更加智能的旅游攻略。通过集成 Chroma 向量数据库和 DashScope 文本嵌入模型，构建了本地旅游知识库。Agent 能够利用 RAG 机制，从 PDF 等非结构化文档中提取并利用相关知识，有效提升了问题回答的准确性，在一定程度上缓解了 LLM 的幻觉问题，同时还为 Agent 设计合理的 Prompt，引导 Agent 生成了更合理的回复，为出行游客提供了个性化的问答服务。

参考文献

[1] 李广丽,朱涛,袁天,等.混合分层抽样与协同过滤的旅游景点推荐模型研究[J]. 数据采集与处理,2019,34(3):566-576.

[2] ACHIAM J, ADLER S, AGARWAL S, et al. Gpt-4 technical report [J]. arXiv preprint arXiv:230308774, 2023.

[3] Azamfirei R, Kudchadkar S R, Fackler J. Large language models and the perils of their hallucinations[J]. Critical Care, 2023, 27(1): 120.

[4] Huang Y, Xu J, Lai J, et al. Advancing transformer architecture in long-context large language models: A comprehensive survey[J]. arXiv preprint arXiv:2311.12351, 2023.

[5] WEI J, WANG X, SCHUURMANS D, et al. Chain-of-thought prompting elicits reasoning in large language models; proceedings of the Proceedings of the 36th International Conference on Neural Information Processing Systems, New Orleans,LA,USA, F, 2024 [C]. Curran Associates Inc.

[6] Li J, Yuan Y, Zhang Z. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases[J]. arXiv preprint arXiv:2403.10446, 2024.

[7] Mavroudis V. LangChain[J]. 2024.