



西華大學
XIHUA UNIVERSITY

Proj225-Deepin 系统文档问答机器人

队伍名称：Bit 编织者队

成员：杜欣 袁贤志 严建滨

指导老师：刘克剑

赛题老师：吴荣杰

2024 年 7 月

目录

1. 赛题说明	3
2. 当前项目完成情况	4
3. 系统框架设计	5
3.1 总体框架介绍	5
3.2 使用的工具介绍	5
LangChain	5
PineCone	5
阿里云 API 大语言模型	6
Streamlit	6
4. 系统实现	7
4.1 数据收集与处理	7
4.2 向量化与索引	8
4.3 网站创建	8
4.4 后端逻辑	8
5. 碰到的问题和解决方法	9
问题 1：选择本地模型微调还是利用大模型 API 完成项目	9
解决策略	9
问题 2：所用模型不能很好得根据 Deepin 中的数据集回答问题	9
解决策略	9
6. 分工和协作	10
项目开发关键环节	10
协作策略	10
7. 提交的仓库目录和文件描述	11
8. 收获和心得	12

1. 赛题说明

- 项目名称：文档问答机器人

- 项目描述：

<https://wiki.deepin.org> 上有 900 多条 deepin 系统相关的中文教程和词条，请编写能根据这些内容回答问题的中文聊天机器人。使用者通过命令行界面输入问题，机器人输出回答和参考的 wiki 文档的链接。

- 项目预期目标：

（必须）完成一个聊天机器人，能根据 deepin wiki 内容回答问题

（必须）通过训练模型，使回答和问题要有 80%以上概率的相关性。

（必须）编写博客，记录开发过程的心得与体会，并将博客投递至 planet.deepin.org

（可选）添加 <https://linglong.dev/> 使用手册内容，支持回答玲珑使用的问题。

（可选）使用 web 或 qt 为机器人添加可视化界面支持，能在 deepin 系统上使用。

- 参考资料：

问答系统 wiki: <https://zh.wikipedia.org/zh-cn/问答系统>

问答系统用例: <https://huggingface.co/tasks/question-answering>

deepin wiki 仓库:

<https://github.com/linuxdeepin/wiki.deepin.org>

2. 当前项目完成情况

目标内容	完成情况
完成一个聊天机器人，能根据 deepin wiki 内容回答问题	完成。开发了一个基于大模型的问答机器人，目前能够大致理解 deep_wiki 和玲珑使用手册的内容并回答问题。
通过训练模型，使回答和问题要有 80%以上概率的相关性。	部分完成。暂时没想到怎么进行定量分析。
编写博客，记录开发过程的心得与体会，并将博客投递 planet.deepin.org	完成。已编写和投递博客。
添加 https://linglong.dev/ 使用手册内容，支持回答玲珑使用的问题。	完成。已通过爬虫技术爬取玲珑使用手册内容，并已经存入向量数据库。
使用 web 或 qt 为机器人添加可视化界面支持，能在 deepin 系统上使用。	完成。使用 Streamlit 库搭建了简约界面，可以在本地浏览器进行可视化使用。

3. 系统框架设计

3.1 总体框架介绍

本项目使用 RAG 架构，通过在向量数据库中查询和问题最相关的几个文本，将问题和相关知识输入大模型从而得到基于自有知识库的问题回答，这样做可以使得机器人更擅长特定领域的回答。项目的技术路线框架图如 3.1 所示。

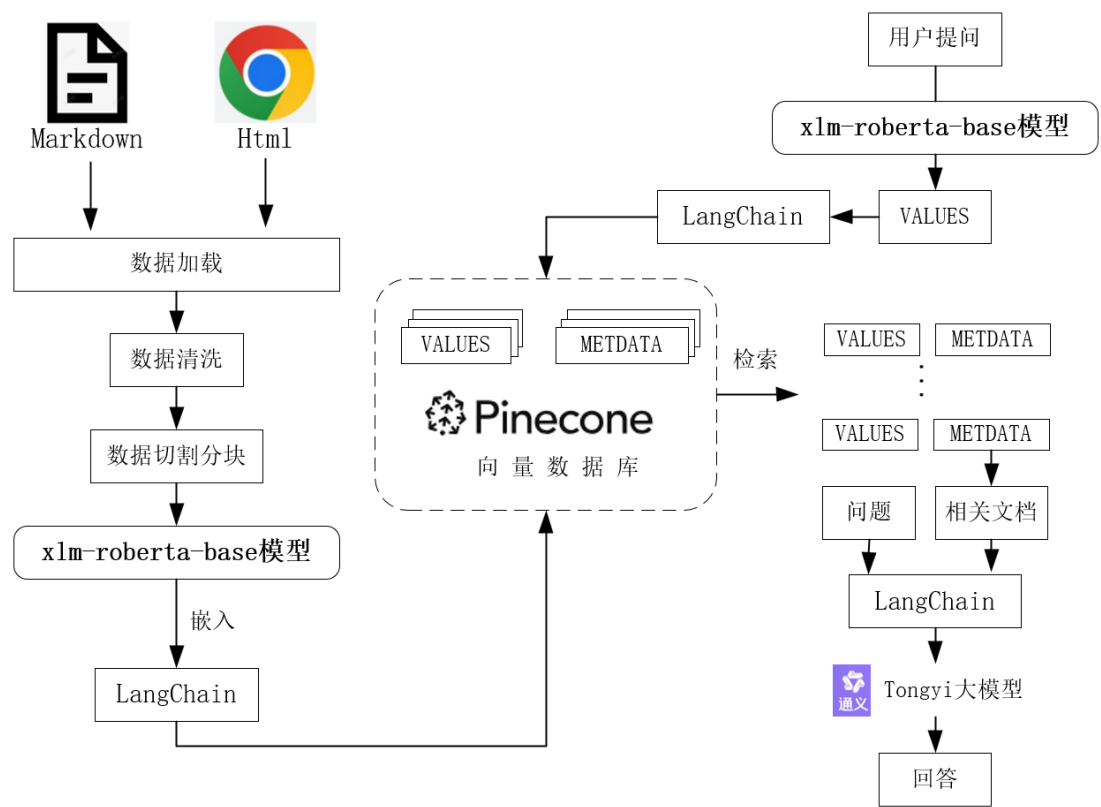


图 3.1 系统框架图

3.2 使用的工具介绍

LangChain

LangChain 作为一个粘合剂，帮助我们将不同的工具和组件整合到我们的问答机器人中。它将数据、向量数据库和大型语言模型 (LLM) 紧密结合，自动化工作流程，优化信息处理，并确保在整个问答过程中提供连贯、准确和定制化的用户体验。

PineCone

PineCone 是一个向量数据库，专门用于处理和存储向量化的数据。在我们的项目中，PineCone 将用于存储和检索 deepin wiki 和玲珑使用手册的向量化表示，以便于快速检索和回答。

阿里云 API 大语言模型

阿里云 API 将提供强大的语言处理能力，帮助分析文档内容，构建问答对，并训练问答模型。在本项目中，我们使用 Tongyi 大型语言模型根据检索到的文档生成答案，即结合知识库利用 embedding 而不是大模型微调的方式实现答案的生成。

Streamlit

Streamlit 是一个开源工具，用于快速创建和分享数据应用程序。我们将使用 Streamlit 来构建机器人的 Web 界面，提供一个简单易用的用户交互平台。

4. 系统实现

4.1 数据收集与处理

下载和爬取 <https://wiki.deepin.org> 上的 900 多条 deepin 系统相关的中文教程和词条以及玲珑使用问题。二者都是以 .md 或 .html 文件的形式存储的，部分数据如图 4.1 所示。

利用 langchain 自带的 UnstructuredMarkdownLoader 和 UnstructuredHTMLLoader 工具方便的加载数据文件，将其放至列表中以便处理。

由于不同的数据格式，原始文本中存在着各种特殊字符、标点符号、数字、链接、HTML 标签等噪声因素，这些因素会对文档问答的效果产生负面影响。因此，在加载文档数据后，先进行文本数据的预处理。使用正则表达式进行文本清洗。去除 html 标签、特殊符号和干扰数据。它通过描述字符组成和字符之间的关系，识别、提取和替换文本中的特定模式。

```
title: Commit 提交信息规范
description: 关于 deepin 源码仓库提交时的提交信息规范
published: true
date: 2023-02-22T09:12:39.635Z
tags:
editor: markdown
dateCreated: 2022-08-30T03:18:16.527Z
```

为保证 git 提交记录能够清晰的反应提交的内容，我们制定了如下 commit 提交信息规范。此规范涵盖了社区贡献者与 Deepin 内部均需遵守的规范格式。

由于社区贡献者与 Deepin 内部员工的贡献方式有别，关于以下规范中的部分信息是否需要填写的要求也会有所差异，将会在随后的段落中另行描述。

请注意，Commit 提交规范适用于 git commit 时所填写的提交信息，
而不是在 GitHub 发起 Pull Request 时所填写的标题与描述。
{.is-warning}

Commit 格式小抄

以下为满足目前提交信息规范的基准格式。

```
类型：标题概要

详细描述
```

图 4.1 文档示例

因为大模型支持处理的文字字数有限，因此我们对文档进行分割处理，分割后的文档最大字数设置为 256，将所有文档分割好后放至列表中用于下一步向量化。

4.2 向量化与索引

将预处理后的文本数据用 SentenceTransformer 库的 xlm-roberta-base 模型进行嵌入向量化，并且存入 pinecone 中，使用 PineCone 创建索引，以便快速检索。如图 4.2 所示为文档在切割后存入 Pinecone 的示例，包含它的 VALUES (即向量) 和它对应的文本。

	ID	VALUES	
1	00acc4f4-66...	-0.0377608351, 0.030005591, 0.0285773966, 0.00115396769, 0.0161107201, -0.0483232439, 0...	
METADATA			
text: "Algorithm (信息-摘要算法) 5, 它是一种不可逆的加密算法。软件或文件一般都有自己固定的文件格式或信息，简单一点说就是"世界上没有完全..."			

图 4.2 文本在 Pinecone 中储存的示例

4.3 网站创建

用 Streamlit 快速创建动态的数据驱动界面，实现实时数据展示和用户交互。设置好输入框，当用户输入问题开始搜索时，启动 Pinecone，同样使用 SentenceTransformer 库的 xlm-roberta-base 模型对用户问题进行向量化，用 pinecone 基于 query 和相似度在向量库中进行查询，使用 Tongyi 大型语言模型生成答案，并将参考的文档和结果一并显示到网站页面。问答界面如图 4.3 所示。



图 4.3 问答系统界面

4.4 后端逻辑

LangChain 将作为后端逻辑的核心，处理用户请求，调用阿里云 API 进行文本分析，从 PineCone 检索答案，并通过 Streamlit 展示结果

5. 碰到的问题和解决方法

问题 1：选择本地模型微调还是利用大模型 API 完成项目

解决策略

研究比较：深入研究并比较两种方案的优劣，包括技术实现的难易程度、成本效益分析、以及未来的可维护性和扩展性。

评估决策：鉴于当前缺乏高显存硬件资源，本地模型微调存在实施难度，因此决定采用调用现有大模型 API 的方式，以确保项目的顺利进行和高效运作。

问题 2：所用模型不能很好得根据 Deepin 中的数据集回答问题

解决策略

数据质量审查：分析并审查数据集的预处理流程，确保数据的质量，包括格式的一致性、数据的清洗以及标注的准确性。

优化数据处理：调整和优化数据预处理步骤，以提高数据集的质量，从而提升模型在特定数据集上的表现和准确性。

6. 分工和协作

本次项目开发主要涉及到的部分有：数据获取和处理、将文档数据嵌入到数据库、编写主程序实现问答功能、利用 streamlit 开发可视化界面、技术文档的撰写、编写博客等。在协作方面，我们团队通过定期的线下项目会议、在线协作平台等方式，确保信息的及时传递和共享。各团队成员之间紧密配合，共同推进项目的进展。

项目开发关键环节

数据获取和处理：负责收集所需数据，并进行必要的清洗、转换和标准化处理。

数据库集成：将处理后的数据有效嵌入数据库，确保数据的可查询性和高效存取。

核心功能开发：编写主程序，实现项目的问答核心功能，确保算法的准确性和响应速度。

用户界面设计：使用 Streamlit 等工具开发可视化界面，提升用户体验和操作便捷性。

文档撰写：负责技术文档的编写，确保项目的技术细节和实现过程有详尽记录。

知识分享：通过编写博客等形式，分享项目经验和技術洞见，促进知识传播。

协作策略

定期沟通：通过定期的线下项目会议，确保团队成员间面对面的深入交流和问题解决。

在线协作：利用在线协作平台，如项目管理工具、代码仓库等，实现信息的即时共享和远程协作。

角色明确：明确每个成员的职责和任务，确保每个人都清楚自己的工作重点和交付目标。

进度同步：通过项目管理系统跟踪进度，确保团队对项目的整体进展有清晰的认识。

7. 提交的仓库目录和文件描述

提交的仓库如图 7.1 所示。

/data 文件夹是存储的 deep_wiki 和玲珑使用手册的原始文件。

.env 是储存 api_key 的环境变量。

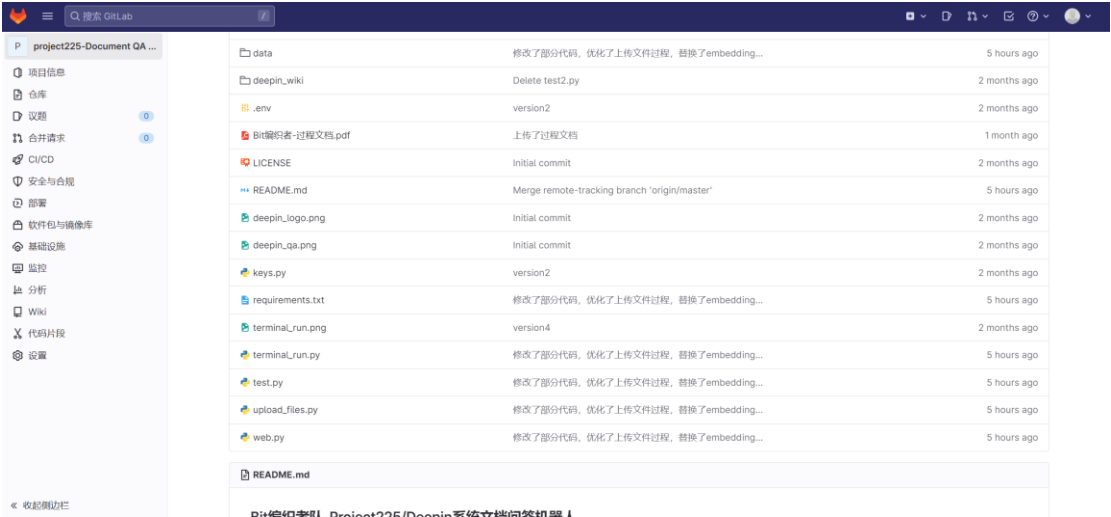
Bit 编织者-过程文档.pdf 是本次项目的技术文档，即本文档。

terminal_run.py: 在终端以 python terminal_run.py 运行。

test.py:用于测试功能。

upload_files.py: 用于处理和上传文档到 pinecone。

web.py: 在终端使用 streamlit run web.py 即可实现在本地浏览器进行可视化使用。



7.1 仓库截图

8. 收获和心得

本次比赛不仅让我们在学习技术上有了进步，也让我们在团队协作上获得不少的收获。

1. 在比赛中深入理解了 RAG (Retrieval-Augmented Generation) 框架的工作原理，特别是在处理开放域问答任务时的优势。通过 LangChain 框架，我们成功地将多种语言模型和服务连接起来，简化了开发流程并实现了模型之间的无缝集成，从而显著提高了问答机器人的回答质量和准确性。我们利用 Pinecone 作为向量数据库来存储和检索文档的嵌入表示，这极大地提高了检索速度和精确度。通过优化向量索引，我们能够快速找到与查询最相关的文档片段，进而提升问答机器人的响应效率和用户体验。
2. 在整个比赛过程中，我们深刻体会到了团队合作的重要性。通过有效的沟通和分工合作，确保了项目的顺利进行。每个成员的专业技能互补，共同克服了项目中的技术难题，为我们今后的合作奠定了良好的基础。通过在有限的时间内完成项目，我们学会了如何优先处理关键任务，并有效管理时间。当遇到技术挑战时，我们通过在线资源、社区讨论和技术支持等多种途径寻求解决方案，锻炼了我们的解决问题的能力，养成了持续学习的好习惯。