南开大学实习实训项目说明书

8 组-八大菜系菜谱知识图谱智能问答系统项目说明书

作者：傅桐 & 彭钰钊 & 赵康明 & 张弛

组织：中软国际（人工智能方向）实习实训八组

时间：July 20, 2023

版本：8.0

南開大學 Nankai University

团队宣言：落地为兄弟，何必骨肉亲！得欢当作乐，斗酒聚比邻。

目录

1 项目概述

1.1 简介

* **项目名称: 老八食谱**

本项目由不被优化小组倾情制作。

1.2 目标与背景

南开大学有着光荣的爱国传统。这份爱也体现在对祖国的传统文化的热爱上。

中华文化源远流长，博大精深。在绵延不绝的文化长河中，丰富多彩，令人垂涎的各种中华美食必然是其中的一颗明珠。

为了结合计算机技术，让我们大家都非常喜爱的美食文化生动的展现出来，我们选择了这个课题，旨在通过我们收集和处理数据，为大家清晰的呈现中华美食的丰富多彩，加深大家对于中华美食的理解和喜爱，并且将其相关知识深入脑中。

2 项目组成

本项目的呈现分成两个部分。

- **第一部分——知识图谱可视化**

  第一部分是对我们已经存入设计好知识图谱的 Neo4j 数据库进行可视化呈现，旨在通过清晰的展现各大菜系之间的菜品和食材的联系，并且辅以精美的网页界面和音乐带给用户视听双重刺激来加深用户对于中华传统美食的理解与印象。

- **第二部分——智能问答系统**

  第二部分是基于上述数据库的数据，进行智能化的问答系统的构建。目的是通过与系统进行有趣的智能的问答，在获得经由系统筛选挖掘的知识的同时，加深对于中华传统美食的记忆与认知。

3 数据库设计

3.1 数据库的选择

我们希望做到的是构建多个菜系菜品和食材之间的联系，为此我们选择的是 neo4j 图数据库。只需要为不同的菜系、菜品和食材（原料）构建结点，然后将它们用关系联系在一起。

3.2 数据格式的设计

- **菜系**

  对于菜系，拥有属性菜系名。

- **菜品**

  对于菜品，拥有属性菜品名，口味 工艺 耗时 难度，制作步骤。

- **原料**

  对于原料，拥有属性原料名。

- **联系**
  - 菜品和菜系间具有原料，为（菜品）-[属于]->（菜系）
  - 原料和菜品间具有联系，为（原料）-[用量]->（菜品）；其中用量是一种动态关系，具有表示具体用量的属性值

3.3 数据的爬取

设计好表的结构以后，我们拥有了数据存放标准，便可以开始进行数据爬取的工作。爬取上，我们采用的是 urllib、requests、$BeautifulSoup4$、$lxml$、$requests-html$、$selenium$ 多种库的动静结合的花样繁多的爬取方式，以适应网站不同类型的文本、图像等数据的爬取需求。并且能够实现更加高效的自动化爬取

3.4 数据的预处理

根据从网页获取的原数据（已经经过预先的格式设计和处理处理）采用 pandas 等工具对数据结构进行处理以适应后期导入数据库的需求。

3.5 数据库选择

数据库采用的是Neo4j。在处理这种类型的数据时，图数据库非常的清晰明了。在与python连接时使用 py2neo 库。

4 需求分析

4.1 用户需求

我们通过模拟用户需求，即以用户的视角分析自己的诉求，得到如图1所示的分析结果

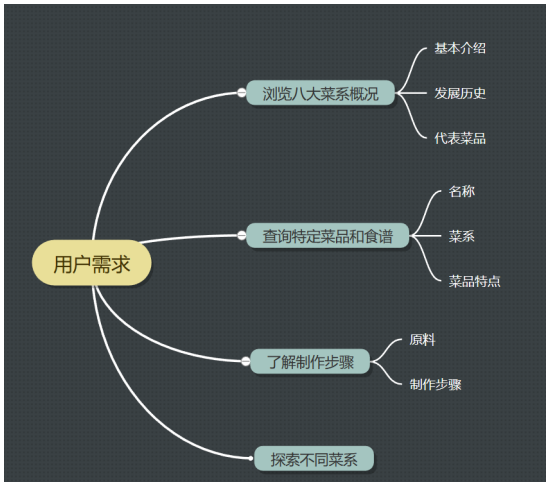

图 1: 用户需求

4.2 功能需求

根据用户需求进行分析我们得到如下功能需求：

- **食谱搜索功能**

  能够通过对于名称的搜索定位菜品以及菜品的相关信息。

- **食谱详情展示**

  能够在可视化界面当中呈现出食谱的详情页。

- **分类浏览功能**

  可以通过菜系搜索不同的菜品。

- **菜系概况展示**

  能够展示菜系的基本信息、历史文化等信息。

- **用户友好界面**

  界面应该具有可交互性和趣味性，能够使用户具有更好的体验。

- **知识图谱可视化界面**

  项目关键之一应该能够展示各种节点之间的关系以及给出相关数据展示。

- **智能问答系统**

  能够实现关于食谱的基本问答功能，给出所需要查询的信息。

5 技术实现

5.1 数据获取

- **概述**

  在对于静态数据的爬取时，直接使用 requests、bs4、lxml 等库直接进行爬取

- **难点处理——动态数据**

  在面对动态页面时，可以利用 requests-html 或者是对页面发送的连接进行规律的总结，然后进入页面请求的信息页进行爬取，以及在处理更加麻烦的动态数据时，采用 selenium 对元素截图与 opencv+pytesseract 对截取到的图片进行处理然后返回具体的字符串。

- **难点处理——元素定位**

  要准确匹配原料和对应的用量是本次数据获取的一个难点，由于存在两种情况——有链接原料标签和无链接原料标签同时用量均无链接，因此需要分两种情况考虑，这也导致可能出现原料和用量不匹配的情况。

  为了解决这一个问题，分析网页结构，我们采用了先定位原料标签 ($category_s1$)，然后根据已经定位的原料找到其 li 结构，然后在匹配同级下的 $category_s2$，经过这样的操作，我们成功获取了原料和用量准确匹配的数据，代码如下：

Listing 1: 元素定位

```
main_ingredient_dict = {}
main_ingredient_names = selector.xpath('/html/body/div[5]/div/div[1]/div[3]/
    div/fieldset[1]/div/ul/li/span[1]/a/b')
for i in range(0,len(main_ingredient_names)):
    main_ingredient_name = main_ingredient_names[i].text
    main_ingredient_dosage = selector.xpath('//li/span[contains(., "{}")]/../
        span[@class="category_s2"]/text()'.format(str(main_ingredient_name)))
    main_ingredient_dict[main_ingredient_name] = main_ingredient_dosage[0]
# 这里是处理没有链接的那种情况，两种定位不一样
main_ingredient_names = selector.xpath('/html/body/div[5]/div/div[1]/div[3]/
```

```
    div/fieldset[1]/div/ul/li/span[1]/b')
if len(main_ingredient_names) != 0:
for i in range(0,len(main_ingredient_names)):
    main_ingredient_name = main_ingredient_names[i].text
    main_ingredient_dosage = selector.xpath('//li/span[contains(., "{}")]/../
        span[@class="category_s2"]/text()'.format(str(main_ingredient_name)))
```

5.2 知识图谱可视化

5.2.1 网页框架

网页框架采用的是 **Django**

Django 提供了许多内置的功能和模块，可以极大地简化开发过程和提高开发效率。而且因为是 python，所以在入门成本上相对便宜。

5.2.2 html 编写

编辑器采用的是**记事本**

记事本的好处是轻便，而且电脑自带。坏处是它是记事本。用起来比较不方便，但是不依靠 IDE 的编程可以极大地锻炼能力。

5.2.3 可视化工具

可视化数据工具采用的是 **D3**

它非常灵活，提供了多组 api，支持多种可视化的图像类型，具有极强的兼容性。最重要的是有非常丰富的社区支持，即使是新人也可以在并不精通的情况下速成页面设计。

5.2.4 系统架构

架构的大体示意图如下：

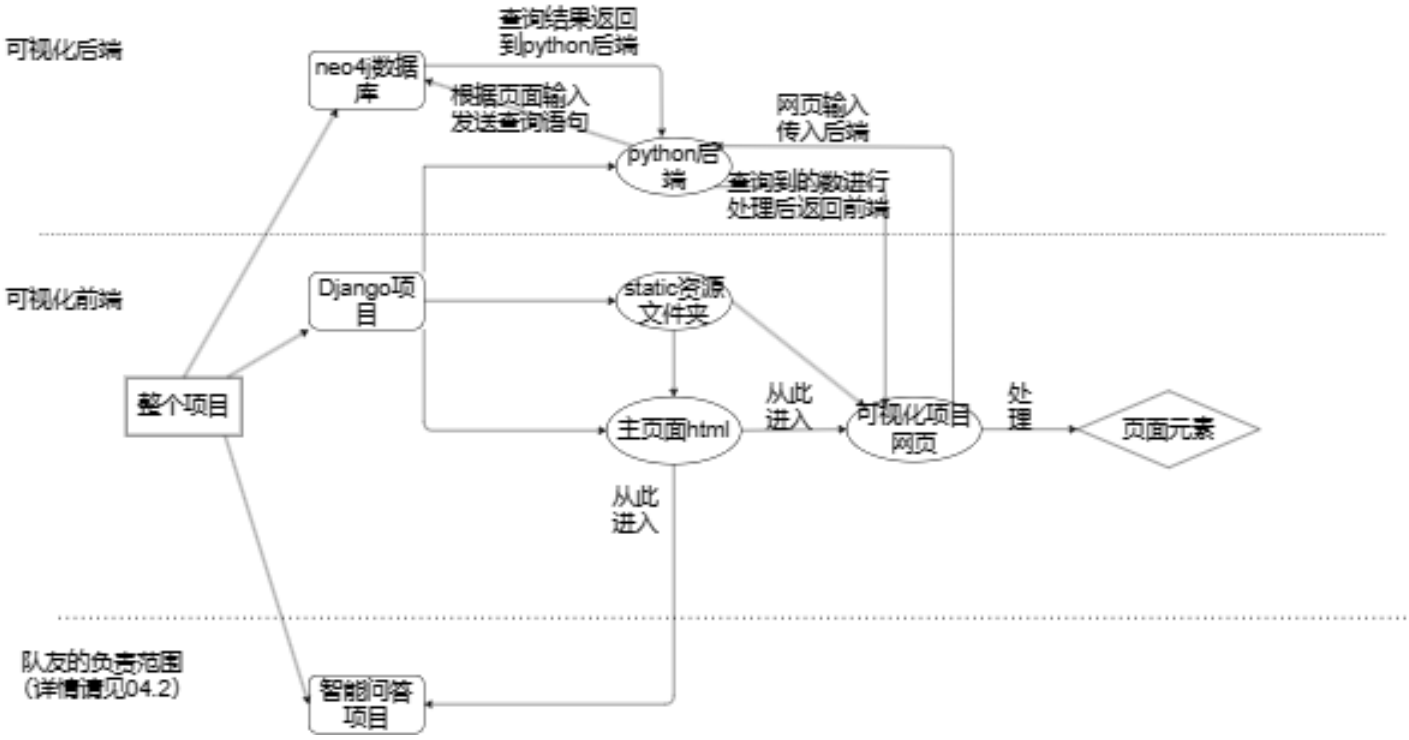

图 2: 可视化系统架构

在这里，后端的主要操作是 python 与 neo4j 数据库的交互，以及 python 对查询结果的处理。

为了保证结构严谨，前后端交互仅预留一个接口，进行我们需要展示的数据的传输工作

前端就对传进来的数据进行可视化操作，并且利用 static 目录下各种资源文件进行页面美化操作，添加各种可互动的功能，提升用户的使用体验与交互感。

5.3 问答系统

5.3.1 数据处理

本着多多益善的探索原则又对基于 fuseki 的 SPARQL 查询进行了学习并将其运用到了 KBQA 中作为云端数据的来源（可是又得搞大量的 RDF 数据，十分痛苦）。

这里主要要做的工作就是将之前处理完成的 csv 数据处理为 APache Jean Fuseki 服务器支持的三元组格式，自定义前缀，并且分表处理最后合并成.nt 文件再导入到服务器中即可，过程中具体的处理代码如下 (选取处理部分表的代码为例):

```
with open(csv_file, newline="", encoding="utf-8-sig") as file:


    reader = csv.DictReader(file)
    for row in reader:
```

```
        if ' ' not in row['菜品名']:
            subject = f"<http://kg.course/ai-food-time/{row['菜品名']}>"
            triples = [
                (subject, "<http://kg.course/ai-food-time/特色>", f'"口味: {row["口味
                    "]}"'),
                (subject, "<http://kg.course/ai-food-time/特色>", f'"工艺: {row["工艺
                    "]}"'),
                (subject, "<http://kg.course/ai-food-time/特色>", f'"耗时: {row["耗时
                    "]}"'),
                (subject, "<http://kg.course/ai-food-time/特色>", f'"难度: {row["难度
                    "]}"'),
                (subject, "<http://kg.course/ai-food-time/口味>", f'"{row["口味"]}"'),
                (subject, "<http://kg.course/ai-food-time/工艺>", f'"{row["工艺"]}"'),
                (subject, "<http://kg.course/ai-food-time/耗时>", f'"{row["耗时"]}"'),
                (subject, "<http://kg.course/ai-food-time/难度>", f'"{row["难度"]}"'),
            ]
            rdf_data.extend(triples)
        else:
            pass
# Convert RDF triples into N-Triples format
nt_data = [f"{triple[0]} {triple[1]} {triple[2]} ." for triple in rdf_data]
# Save N-Triples data into an NT file
nt_file = r"C:\Users\咆哮的小清新\Desktop\实习实训\Final_Project\Cuisine\code\
    m_ntriples.nt"

with open(nt_file, "w", encoding="utf-8") as file:
    file.write("\n".join(nt_data))
```

5.3.2 后端开发

从后端来说，从需求出发，要实现的是输入提问语句然后给出相应回答的功能模块，很容易想到要用到 NLP 的相关方法，而且由于设想中并不需要实现多么智能的人机交互，仅需对预设的几种问题模板作出正确的识别和响应即可，因此没有考虑深度学习等需要海量数据训练模型的方法，经过一番查阅资料，最终敲定了正则表达式模板匹配这种相对而言较为轻

量化的方法。

三元组数据处理完毕并上传到服务器后，接着便是测试根据问题模板解析提问语句、生成 SPARQL 查询语句是否成功，反复调试后也能正常运行，然后事情就开朗了起来，仅需要把对应的查询语句提交到服务器中执行返回结果，解析结果三元组，分类讨论答案的情况便可以输出了，然后就是测试样例反复轰炸，查缺补漏并完善反馈错误机制，到此，后端的实现就基本告一段落了。

具体的代码实现见源码中的 KBQA 模块，此处就不详细介绍了

5.3.3 网页框架

网页框架采用的是 **Flask**

Flask 作为一个轻量级的 Python Web 开发工具，其能够帮助开发者快速进行开发

5.3.4 html 编写

html 编写采用的是 **Vscode**

Vscode 作为一款轻量级的编辑器，能够较为轻松的完成 html 的编写。

6 流程说明

6.1 知识图谱可视化

首先是页面结构的设计。我们想要做的是非常有动感的页面，以关系图谱为中心，通过饼状图和柱状图等展现数据分析结果，做到较为完备的数据展现。然后通过对于展现的数据图添加动态效果提高用户的互动感，再辅以背景和页面的各种小元素等来提高用户的视觉体验。而且总的页面所有的元素都时时刻刻在扣紧传统美食的主题。

在代码编写上，采用前后端并进的，深度细粒度并行化的联合作战方式。我们首先先把目光放在大体的主要功能上。在这里分两路进行前端的数据接收和最基本的结点的构建以及后端的数据处理和发送。在基本实现结点的展现以后，一同进行页面的设计和细节功能的完善。期间需要细心地进行版本控制与岔路的融合。最终工作效率得到了极大的提升。在实现设计时，也不只是完完全全按照设计来，一些过于理想的设计可能导致程序性能下降。这是后就不得不进行性能和效果的权衡，或者是寻找引入类似缓存、预加载、减小资源文件的大小之类的操作。

我们的设计总体上得到了很好的完成。基本实现了一个极具动感的，充满交互性的，拥有了相对完备、广度深度兼具的数据展现功能，并且使得网页“声情并茂”——拥有精美的壁纸、细节以及动听的、切合文化主题的背景音乐。

6.2 问答系统

智能问答系统的大致工作流如下：

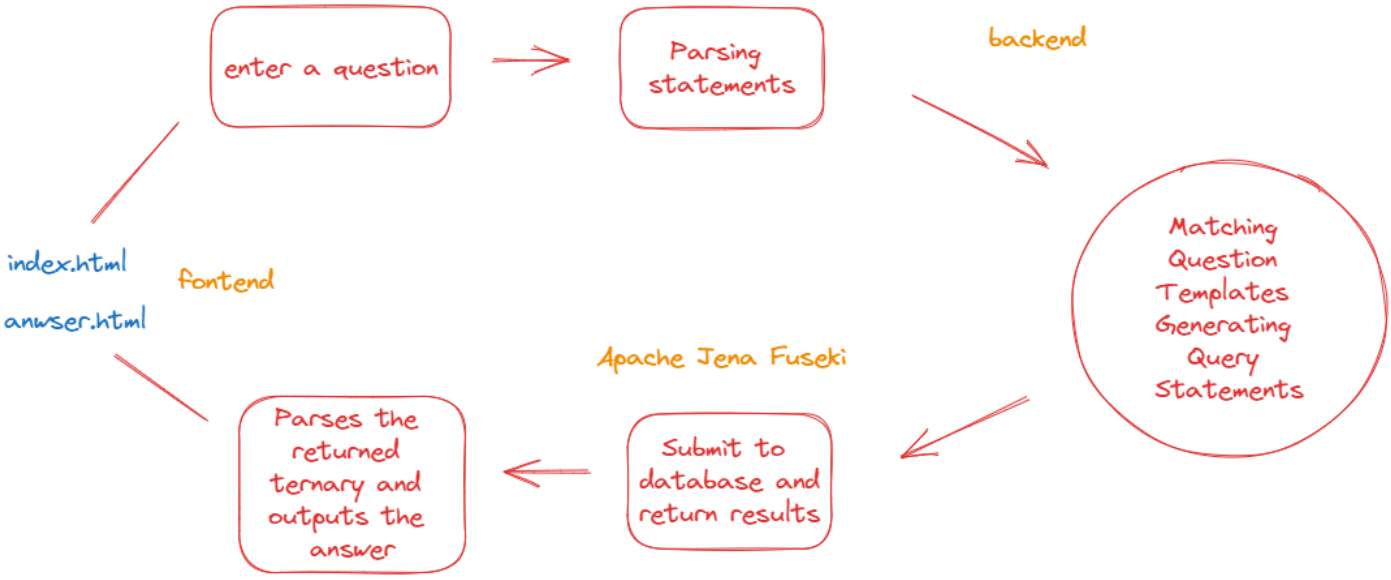

图 3: KBQA 工作流程图

如上图所示即为整个 KBQA 系统前后端交互的流程逻辑，即在前端网页输入问题，然后在后端解析提问并与预先定义的问题模板进行匹配以生成相应的 SPARQL 语句，再将其提交到对应的 APache Jena Fuseki 服务器中执行得到三元组形式的查询结果，分情况解析查询结果然后输出对应答案到前端。

7 项目配置说明

需要运行本项目，您需要按照如下步骤进行环境配置：

7.1 项目文件夹结构介绍

主目录里面的 apache-jena-fuseki-4.9.0 文件夹是智能问答系统的环境支持。在 Cuisine 文件夹下有四个文件夹，其具体内容见每个文件夹内的 README.md。

7.2 数据库

要使用我们项目，您需要安装 Neo4j。执行 neo4j stop 关闭数据库后，将我们的 neo4j.dump 文件放在一个您可以非常容易找到的路径。如果您未更改配置环境变量，那么请在包含 neo4j-admin.exe 的目录下执行：

```
neo4j-admin database load --overwrite-destination --verbose --from-path=your\
    _path neo4j
```

此举应当能够将我们的数据库加载。请注意 linux 和 windows 下的路径有用/和反斜杠的区别

7.3 智能问答系统（含可视化页面）

运行主体项目需要三步，首先在 apache-jena-fuseki-4.9.0 下进入 cmd 命令行，运行如下命令，启动 fuseki 框架，连接到 APache Jena Fuseki 数据库。

```
java -jar fuseki-server.jar
```

然后在 Cuisine->KBQA 目录下进入 cmd 命令行执行如下命令，启动主页面的 Django 项目。

```
python manage.py runserver 0.0.0.0:8080
```

最后在上一步骤的同级目录下运行 app.py 文件启动问答系统，并访问如下项目网址即可。

```
http://127.0.0.1:8080/recipe_knowledge_graph/
```