# 定福庄周边美食推荐与可视化系统

## 一、研究目标

在日常生活中，大学生们时常面临选择美食的困扰，尤其是在面对众多琳琅满目的选择时，“选择恐惧症”使得决定变得尤为困难。针对这一普遍现象，本项目设计并开发了“定福庄周边美食推荐与可视化系统”，旨在模拟真实餐饮应用的推荐机制，帮助用户克服选择难题。该系统基于用户的过往用餐记录及即时需求进行个性化推荐，通过分析历史数据来理解用户的饮食偏好，结合当前的特殊要求（如用餐场景、关注点等），精准定位适合用户的定福庄周边美食店铺。此外，为了进一步辅助决策过程，本项目拟构建一个直观的可视化界面以展示了定福庄周边美食的整体概况，包括店铺的价位分布、评分分布、菜系分布、地理位置分布等，为就餐选择提供更多样丰富的参考。

## 二、系统概览

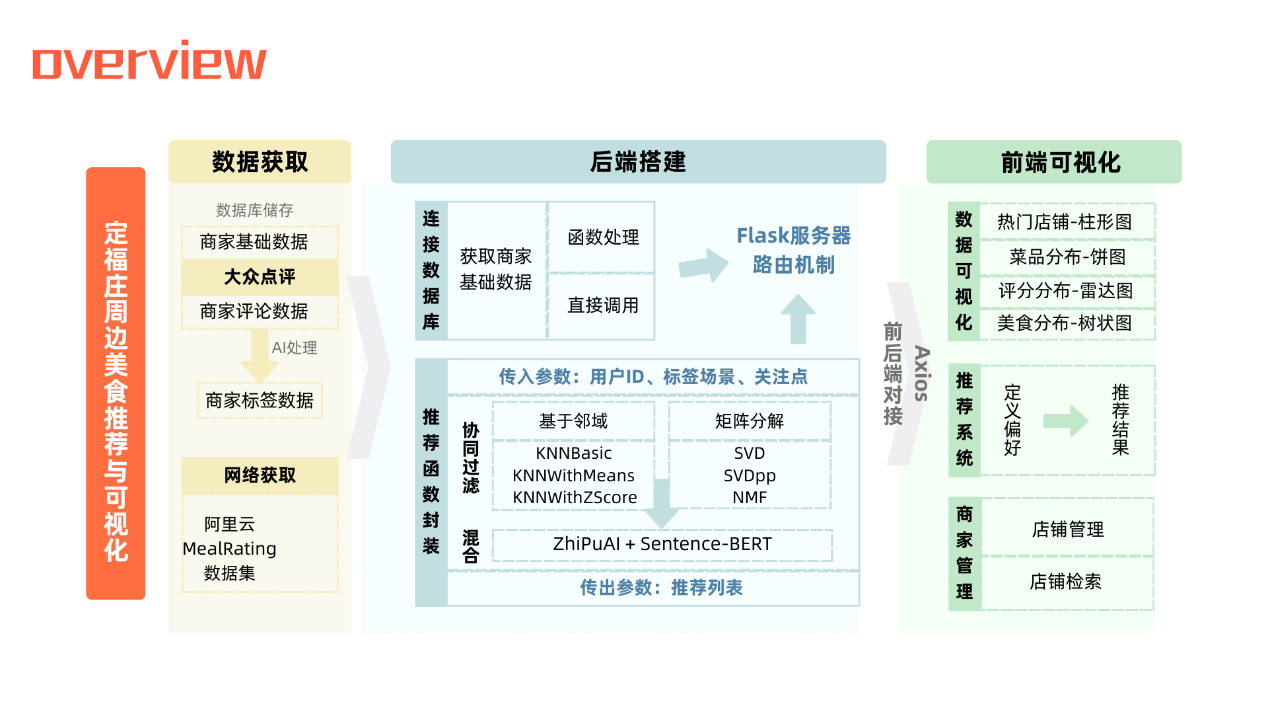


图 1 系统概览图

整体系统搭建主要分为三个阶段：数据获取、后端搭建和前端可视化。数据获取主要通过爬虫和开源社区，并通过AI处理得到标签维度的数据，最终存储到数据库中。

前后端搭建基于VUE组件和Flask服务器，后端连接数据库，获取商家基础数据，一部分进行函数处理后引入Echarts库进行可视化，一部分直接传入前端操作和展示。运用surprise库实现协同过滤算法，并基于内容进行推荐函数设计，在后端封装后，接受前端输入参数并调用推荐函数，传出参数以推荐列表形式展示。

从技术实现层面，系统由三部分组成：基于surprise库实现协同过滤算法、基于内容的推荐函数设计和可视化前后端搭建。其中，协同过滤算法和基于内容的推荐函数都是推荐系统的核心算法，共同作用以生成推荐。可视化前后端搭建是推荐系统的展示层，将算法产生的推荐结果以图形化的方式呈现给用户。

## 三、数据获取

### （一）爬虫获取商家基础信息

**1. 商家信息**

为收集中国传媒大学周边的餐饮数据，选取大众点评作为信息来源平台，“中国传媒大学美食”作为关键词，对搜索结果的前50页数据进行系统性采集，涵盖**商家名称、商家ID、商家链接、特色菜品、位置、评论量、人均消费、地址、电话、综合评分、环境评分、服务评分和口味评分**13个关键指标。

在具体的技术实现过程中：首先运用正则表达式进行数据提取，处理HTML标签和特殊字符；其次，采用专门的JSON解析模块处理评分数据，并使用BeautifulSoup库解析地址和电话信息，所有数据处理函数均采用标准化的列表格式输出结果。系统的核心设计考虑了会话管理的动态性，包括token和uuid等会话相关值的定期更新机制，通过动态获取这些参数，模拟用户通过网页访问商家评分信息时的请求行为，从而得到正确的评分数据。但由于网站的动态特性，代码在地址和电话信息的获取方面遇到了一定的技术障碍，考虑到这些信息并非核心数据指标，最终选择暂时搁置这部分数据的采集。

**2. 评论数据**

在完成对商家信息的数据收集后，通过爬虫进一步收集商家的评论数据，并将其存储到csv文件中。

首先，连接数据库并从中查询商家ID，设定了一系列请求参数，使用user\_agent模拟真实用户浏览器信息行为，通过不断更新网站cookie保证数据的正常获取。为了确保程序不会因为频繁的请求而被网站封锁，设置time.sleep(5)函数来控制请求的频率；其次，对于列表中的商家ID进行遍历，并同样运用多组正则表达式进行数据提取，根据不同的HTML结构提取隐藏和未隐藏的评论及时间信息，并进行数据处理去除特殊字符。在遍历页面请求的过程中，使用了tqdm展现程序的进度，从而能够对程序的运行状态有更直观的了解。最后将获取的信息存储到CSV文件或数据库中，包括了**商家ID、评论时间和评论内容信息**。

### （二）AI获取店铺菜品及其他特色标签

基于此前已获得的店铺评论数据，调用智谱清言AI大模型对评论进行总结。

在提取店铺菜品时，定义extract\_food\_items\_api(comment)函数以接收一条用户评论作为输入参数。同时构建了一个特定的任务提示（prompt），来指导大模型仅识别并返回评论中提及的具体菜品名称，以逗号分隔的列表形式呈现。如果评论中未提到具体菜品，则函数返回空列表。此外，为了确保API响应的有效性，还设置了清理和检查逻辑，能够处理API响应中的异常情况，并适当地解析和分割菜品名称。

在提取店铺场景标签时，定义extract\_features\_api(comment)函数，同样通过prompt指示大模型分析评论中的多个维度，如就餐场景、人群特征、口味特点等，并将这些信息以结构化的JSON格式返回。并且使用正则表达式匹配最外层的大括号及其内容，以保证返回的是有效的JSON对象。最后将文本解析为Python字典，并根据需要返回特征标签。

Prompt示例：prompt = f"""任务：从用户评论中提取具体提到的菜品名称。

要求：

1. 只提取实际点的或吃到的菜品

2. 返回格式为逗号分隔的菜品列表

3. 如果没有提到具体菜品则返回空列表

4. 不要包含额外解释

示例输入："这家店的麻辣烫很好吃，炸串也不错"

示例输出：麻辣烫,炸串

评论内容：{comment}"""

prompt = f"""请分析以下餐厅评论，提取关键特征标签。以JSON格式返回，包含以下类别：

1. 场景标签：就餐场景（如：单人餐、商务宴请、亲子遛娃、家庭聚餐、朋友聚会、情侣约会、主题派对、公司团建、节日聚餐等）

2. 人群标签：消费人群特征（如：学生、白领、家庭、情侣、老年人、亲子等）

3. 口味特征：主要口味特点（如：麻辣、清淡、酸甜等）

4. 就餐时段：用餐时间（如：早餐、晚餐、夜宵、冬天、夏季、工作日、周末、节日等）

5. 特色标签：其他特色（如：出餐快、物美价廉、商场附近、停车方便、不用等位、地道风味、氛围清静、健康饮食、菜量大、服务友好、位置好找、菜品精致等）

场景标签和人群标签返回结果小于等于4个,至少有2个

其他标签只返回评论中明确提到或强烈暗示的标签，没有相关信息则对应类别返回空列表。

返回标签从评论中总结，不一定要从参考中选择

示例输出格式：

{{

"场景标签": ["家庭聚餐"],

"人群标签": ["家庭", "儿童"],

"口味特征": ["清淡", "麻辣"],

"就餐时段": ["夜宵"],

"特色标签": ["环境好", "服务周到"]

}}

### （三）店铺信息整体展示

最终，对于一家餐饮店铺，共可总结出基本信息和特征标签两类属性。基本信息包括商家名称、菜系、消费、评分等，特征标签包括菜品、用餐场景、口味特征、餐厅特色等。这些标签中部分将用于推荐系统设计，与用户信息互相匹配；部分将在该店铺作为最终推荐店铺结果时，进行立体化信息展示，辅助用户做出消费决策。

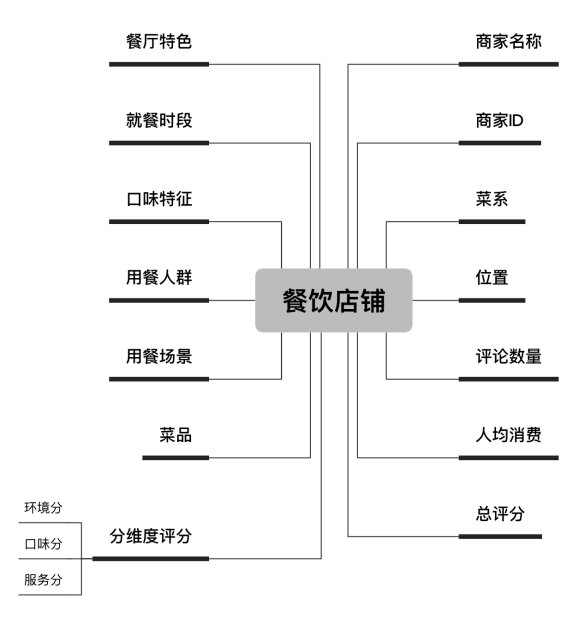


图 2 餐饮店铺数据维度展示图

## 四、系统组成

### （一）系统组成一：基于协同过滤算法的菜品推荐设计

系统组成一基于阿里云MealRating数据集实现了基于协同过滤算法的推荐系统，通过系统化的算法评估、优化和验证过程，成功构建了一个性能稳定、可扩展的推荐系统解决方案。

**1. 数据预处理与基础架构**

网络获取的阿里云MealRating数据集包含5130名用户、1685道菜品和38384条评分记录，评分稀疏度达99.56%。

首先对原始数据进行预处理，使用pandas读取CSV文件并将评分转换为数值类型。通过Reader类，设置评分范围为1-5分，并使用Dataset.load\_from\_df方法构建标准数据集格式。采用train\_test\_split方法将数据按8:2比例划分为训练集和测试集，设置随机种子为42以确保实验可重复性。

**2. 协同过滤算法实现与评估**

系统希望实现多种协同过滤算法并进行性能对比，从而找到最优的实现算法。通过evaluate\_algorithms函数对不同算法进行评估，包括KNNBasic、KNNWithMeans、KNNWithZScore等基于近邻的方法，以及SVD、SVDpp、NMF等基于矩阵分解的方法。评估采用5折交叉验证，使用RMSE（均方根误差）和MAE（平均绝对误差）作为评估指标。检验结果如下表所示，SVD和SVDpp算法表现最优，RMSE分别为0.9753和0.9699，明显优于其他方法。

表 1 模型比较评估

|  |  |  |
| --- | --- | --- |
| **模型** | **RMSE** | **MAE** |
| KNN Basic | 1.1039 | 0.8024 |
| KNN With Means | 1.0628 | 0.7467 |
| KNN With Z Score | 1.0675 | 0.7489 |
| SVD | 0.9753 | 0.7439 |
| SVD pp | 0.9699 | 0.7293 |
| NMF | 1.1074 | 0.8467 |

**3. SVD算法优化与参数调优**

基于初步评估结果，选择SVD算法进行深入优化。通过GridSearchCV实现参数网格搜索，优化三个关键参数：迭代次数在[20, 30, 40]范围内搜索，学习率在[0.005, 0.01]范围内搜索，正则化参数在[0.02, 0.1, 0.4]范围内搜索。最终，optimize\_svd\_params函数通过5折交叉验证确定最优参数组合：n\_epochs=40，lr\_all=0.005，reg\_all=0.1，其对应的最优RMSE为0.9599。

**4. 模型训练与性能验证**

使用train\_final\_model函数实现最终模型的训练和验证。该函数首先使用最优参数初始化SVD模型，在训练集上进行拟合，然后在测试集上进行预测。最终模型在测试集上取得了RMSE为0.9652、MAE为0.7272的性能指标。这个结果虽然高于Netflix Prize竞赛的获奖方案（RMSE为0.8567），但考虑到餐饮推荐场景中个人口味差异较大的特点，这个性能指标仍然是可接受的。

**5. 推荐生成实现**

get\_top\_n\_recommendations函数实现了具体的推荐生成逻辑。该函数首先获取用户未评分的菜品列表，然后使用训练好的模型对这些菜品进行评分预测。预测结果按评分降序排序，选取得分最高的n个菜品作为推荐结果。函数返回包含菜品ID、名称、类型、价格和预测评分的推荐列表。实验结果表明，该方法能够有效处理高稀疏度数据，为用户生成个性化的推荐结果。

**6. 系统性能分析**

在实际应用场景中，系统表现出良好的扩展性和实用性。特别是在处理高稀疏度（99.56%）数据时，基于矩阵分解的SVD算法展现出明显优势。通过参数优化和交叉验证，系统在保持较高预测准确度的同时，也确保了推荐结果的稳定性和可靠性。

### （二）系统组成二：基于内容推荐的函数设计

基于系统组成一算法得出的用户可能喜欢菜品，结合用户输入的场景需求与关注点偏好，为其推荐定福庄周边的美食店铺。其中有两个处理阶段：一是将“用户可能喜欢菜品”与数据集中店铺的菜系、菜品特征进行映射关联，以得出店铺的“菜品相似度分数”，作为最终推荐分数的组成部分；二是设计计算最终推荐分数的函数，将店铺的多维度属性特征与用户信息结合进行推荐。

1. **菜品-店铺映射**

1.1. 菜品描述生成与店铺描述生成：大模型调用

由于算法得出的用户可能喜欢菜品仅有“菜品名称”的输入参数，且菜品名称数据集与店铺菜品数据集完全不相同，因此无法直接进行映射。故调用大模型为菜品和店铺分别生成描述，通过描述文本进行映射。

菜品描述Prompt："请对以下菜品生成一条描述：{comment}。要求：含原料、炒制方法、口感、菜系等，不需要复述菜品名称，以“一道……”开始描述。至少要求50个字，客观描述。示例输入：“小鸡炖蘑菇”，示例输出：一道经典的东北菜，以其鲜美的味道和丰富的营养而广受欢迎。‌这道菜的主要食材包括鸡肉、干蘑菇和粉条，烹饪方法主要是炖煮。鸡肉的鲜嫩与蘑菇的香味结合，使得这道菜口感丰富，味道浓郁。"

店铺描述Prompt："请对以下店铺生成一条描述：{comment}。要求：强调店铺的菜系，以及两道招牌菜的介绍。招牌菜介绍中包括菜系、原料、制作方法、口感，每道菜描述至少40字。示例输入：“烤羊腿老虎菜,骨肉相连,小红茶,手工猪肉大葱水饺,……，烤羊排,大拉皮,烤生蚝,扇贝,香菜,羊大骨棒,烤鳕鱼,皮蛋豆腐,炸串,烤羊肉串”示例输出：该店铺菜系为烤羊腿，招牌菜1是手工猪肉大葱水饺，是东北菜系，原料为猪肉馅、大葱、饺子皮，制作方法是将馅料包入皮中沸水煮熟。口感皮薄馅多，肉香葱脆，汁水丰富。招牌菜2是东北大拉皮，是东北菜系，原料主要是红薯淀粉制的宽粉皮、黄瓜丝、胡萝卜丝等。制作方法是将宽粉皮煮熟后过凉水，配以调料和蔬菜拌制。口感爽滑筋道，清凉可口，适合夏季食用。"

为保证两段文本相似度计算的可行性，做了两个描述文本的生成限定：一是店铺描述文本为店铺菜系加招牌菜介绍，在颗粒度上细致到菜品；二是基于限制一，对两段描述中的菜品描述做了相同格式的限制，均要包括菜系、原料、制作方法、口感维度，以便找到更多共同点。

1.2. 相似度计算：Sentence-BERT模型

对于两个文本的相似度计算，采用Sentence-BERT模型对输入文本进行编码，产生对应的语义嵌入向量。接着利用工具库 util 中的 pytorch\_cos\_sim 函数来计算两个嵌入向量之间的余弦相似度。最后将得到的相似度分数从张量中取出并转换成浮点数，然后乘以10转换为10分制的评分，最后返回这个评分。将该评分保存在每个店铺的属性下，后续计算店铺最终推荐分数时使用。

1. **函数设计**

结合用户已有和输入信息，对于每个店铺，在推荐算法中考虑如下店铺属性：菜品相似度分数、场景相似度分数、关注点相似度分数、总体评分。

菜品相似度分数上，考虑到系统1中会有5个用户喜爱菜品结果输入，故为了在最后推荐结果上同时考虑用户的不同类型菜品偏好，将每道菜品分开取相似度最高店铺。

对于每一菜品下相似度得分最高的20家店铺，限制最终推荐分满分为5分后，对店铺不同属性进行权重设置。

菜品相似度分数：设置权重0.8，对满分10分的菜品相似度乘以0.4系数，满分4分。

场景相似度分数：设置权重0.08，如店铺场景标签包含用户输入场景，加1分，满分0.4分。

关注点相似度分数：设置权重0.1，如店铺在用户关注点的评分表现>4，加1分，满分0.5分。

总体评分：设置权重0.02，如店铺总分>4.5，加1分；大于4.3，加0.5分。满分0.1分。

每一步相似度得分加权得最后店铺推荐分，分别取每道菜品下推荐分最高的1个店铺，得到最终为用户推荐的5个店铺。

### （三）系统组成三：可视化前后端搭建

系统采用基于Vue+Flask框架的前后端开发，使用Axios发送请求和接收数据，实现前后端交互。系统总共包括三个组件：使用Echarts对商家信息进行数据可视化展示、通过前端传入信息调用后端封装推荐算法函数生成推荐列表、连接存储商家信息的sqlite数据库，实现对商家信息的展示与搜索。

1. **前端框架搭建与页面编写**

系统采用Vue.js作为前端开发框架构建用户界面，页面的header部分通过的路由机制为不同的页面分配相应的路径，开发了商家信息可视化页面、美食推荐系统页面和周边商家管理与搜索页面，通过前端的导航设计使用户能够便捷地在不同页面间切换。

1. **Echarts可视化图表**

引入Echarts库构建一个可视化数据页面的布局，通过 el-row 和 el-col 组件进行布局控制，将四个自定义图表组件（柱状图、饼图、雷达图和树状图）按照布局规则排列在页面上。

在图表展现方面，根据评论数选取热度前十的商家，通过柱状图和折线图组合的形式对比热门商家的评分情况；根据商家菜品类型数量分布情况制作饼图；对总评分、环境评分、口味评分和服务评分四个维度绘制展现最大值、最小值和平均值的雷达图；使用树状图展示不同商圈的菜品类型数量分布。

1. **推荐函数封装**

结合系统组成一和二，在后端封装推荐算法函数，定义名为 get\_final\_recommendations 的函数，接收从前端传入的三个参数：UserID、user\_meal\_scene 和 user\_focus\_point，分别代表用户ID、用户的用餐场景标签和用户的关注点，通过协同过滤和基于内容的推荐算法，返回个性化的店铺推荐列表。

1. **周边商家展示与搜索**

使用 Vue 的数据绑定、组件通信和事件处理功能，结合 Element UI 的组件，提供了一个基本的周边店铺管理页面框架，包含了店铺信息的展示、搜索、添加、编辑和删除操作，以及分页功能。通过表格展示店铺的各种信息，用户可以使用输入框搜索店铺、使用分页组件切换页面，并可以通过点击按钮进行店铺信息的编辑和删除操作（目前还未实现保存功能）。

1. **后端搭建**

搭建Flask 服务器处理前端发送的各种请求，根据请求的类型和参数执行相应的业务逻辑。主要通过路由机制，将不同的 URL 请求映射到相应的处理函数上，确保系统的不同功能模块可以正确响应前端的需求。

其中Flask服务器与存储商家信息的 SQLite 数据库相连接，实现对商家信息的读取、存储和更新操作，通过。当接收到前端的数据请求时，它可以从数据库中获取相应的数据并处理，然后将结果返回给前端。

1. **前后端对接**

使用Flask+Axios进行前后端对接，通过Axios库前端可以向后端发送异步请求，进行获取商家信息、请求推荐列表、搜索店铺等操作。后端的 Flask 服务器会根据前端的请求，调用相应的处理函数，并将处理结果以 JSON 格式返回给前端。前端接收到数据后根据数据的类型和用途进行相应的处理，例如更新页面内容、更新可视化图表或刷新推荐列表信息。

### （四）系统存在问题说明

推荐系统封装函数运行需使用VPN，且运行时间过长（>5min），所以向前端返回值时会因超时而无法显示，下图是程序开始运行时，前端会持续加载，最后无法接收返回值，超时显示无推荐结果：

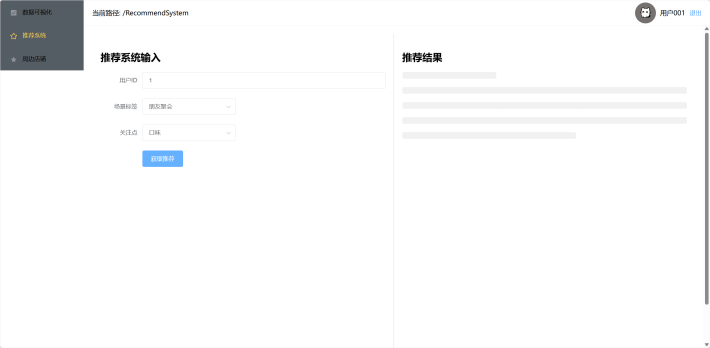


图 3 运行结果1

图形用户界面, 应用程序

中度可信度描述已自动生成

图 4 运行结果2

此时后端会继续运行，直至得到结果：

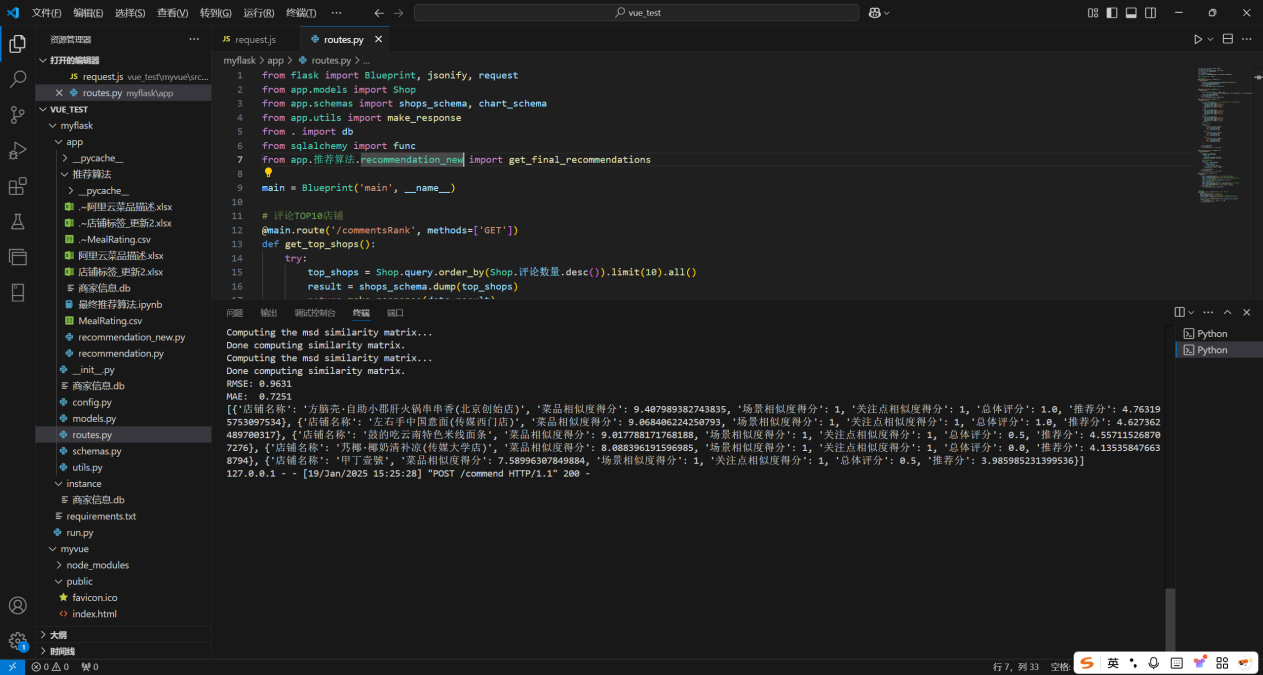


图 5 运行结果3

为了检验是否是前后端未能连接产生的错误，新建recommendation.py（推荐系统封装函数为recommendation\_new.py），函数意味前端传输的三个参数不为空时，返回一个列表（与推荐系统列表完全吻合）。

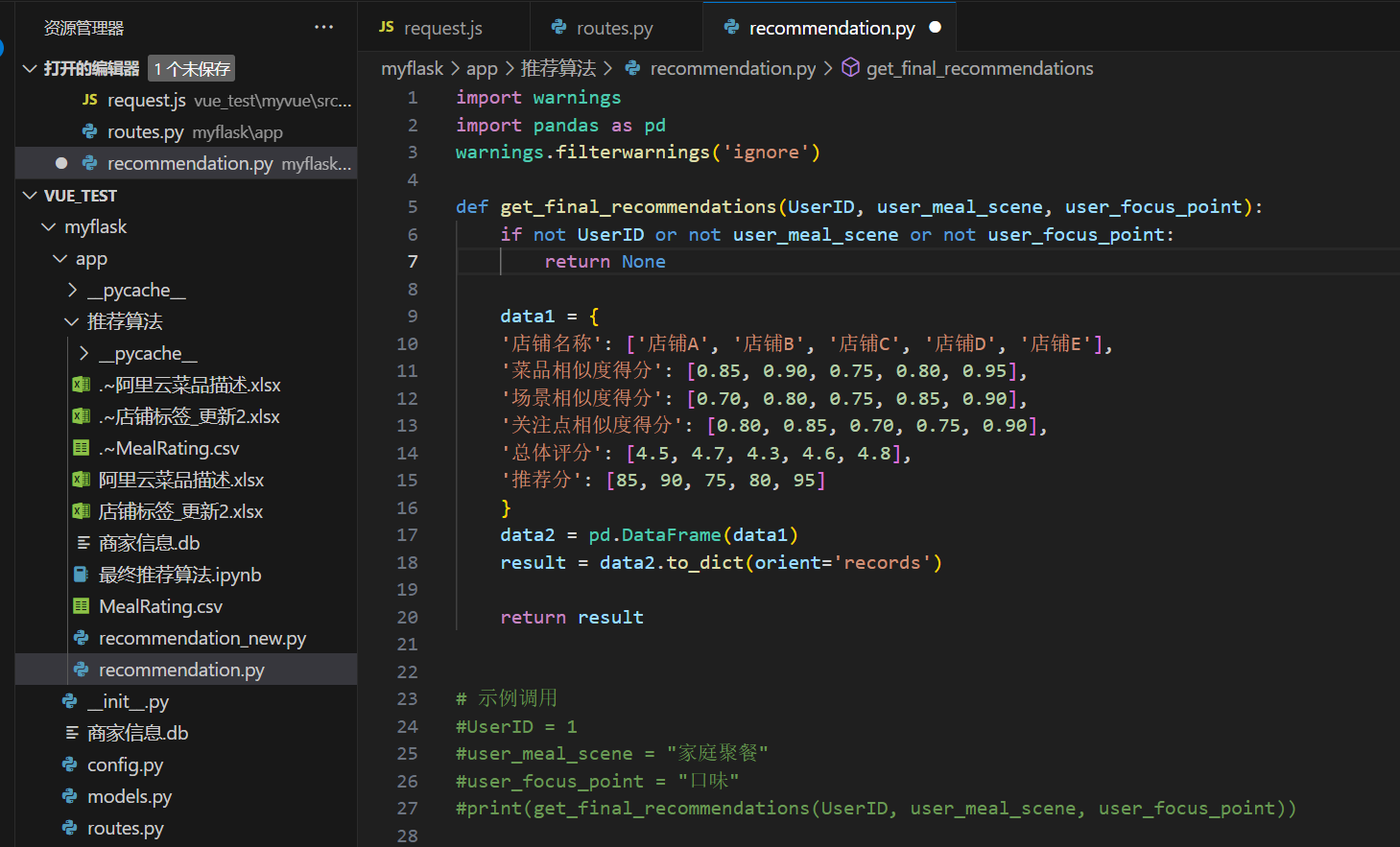


图 6 测试代码

将前后端交互的函数改为简化的测试版，前端能顺利得到并显示结果，证明并非是前后端连接有误。

图形用户界面, 表格

描述已自动生成

图 7 运行结果4

后又采用设置请求超时时间、异步任务处理等方法，均无法解决，故系统仍待优化。

## 五、工作总结

### （一）个人工作

在研究工作中，我们遭遇了许多挫折。在选题阶段，我们本想研究一个新品牌的营销效果与销售转化系统，但是由于需要获取的数据量比较庞杂，具有较高的实现难度，纠结再三后更换为当下的餐饮推荐系统。再者，在使用爬虫获取大众点评评论数据时，由于cookie需要经常更新（学艺不精，爬虫技术水平一般），导致整个爬虫工作非常依靠人力。最终爬虫工作是小组3个人通力合作，耗时多日完成的。

我在研究工作中主要负责爬虫、协同过滤算法、大模型调用的工作，而推荐系统由我和杨同学共同讨论，不断调整得到。在获取大众点评餐饮店铺数据后，由于缺乏“人”的数据，导致研究一度陷入僵局：我们无法获得“人-店”关系数据，若自己设计匹配函数，推荐结果是否合理也难以检验。于是，我开始思考，是否可以找到合适的开源数据对这部分的空白进行补充。最终，我在阿里云平台找到了RealMeating数据集，其包含“人-菜”关系数据，我认为，可以通过“人-菜-店”建立链式传导关系，最终实现餐饮店铺个性化推荐的目的。经过和杨同学的激烈讨论，我们最终商定使用“人-菜-（人）-店”的推荐结构，即首先通过协同过滤得到推荐菜品，再结合“菜-店”关系、“人-店”关系进行相似度运算得到最终结果。我认为该方案是可解释的：首先，人有个性化的饮食口味偏好，在选择在哪里吃的时候，也会首先看这家店有没有自己想吃的东西，所以可以根据饮食数据得到推荐菜品，从“吃什么”的角度定位“在哪吃”的问题；其次，餐饮属于服务业，每个人对服务、环境等有不同的标准和偏好，可以结合店铺本身的属性进行推荐，从“店怎样”的角度确定“在哪吃”的选择；最后，将“吃什么”和“店怎样”结合，可以综合双重角度进行个性化餐饮推荐，从而得到较为可靠的结果。在大模型调用方面，我选择了zhipuai中的开源模型，主要通过prompt得到内容提取、标签数据处理等结果，通过修改prompt和数据存储格式，便可实现不同的处理要求。需要注意的是，数据量越大，大模型的处理时间越长，所以要注意代码运行进度并规划好时间。

在研究和实现基于协同过滤的餐饮推荐系统的过程中，我对推荐系统的构建和优化有了更深入的认识。首先，餐饮推荐场景的特殊性给系统实现带来了独特挑战。数据集的高稀疏度（99.56%）反映了现实世界中用户行为数据的典型特征，即大多数用户只会对少量商品进行评价。这种特征使得传统的基于近邻的协同过滤方法表现不佳，而基于矩阵分解的方法则展现出明显优势。这说明在实际应用中，算法的选择需要充分考虑数据的特征和应用场景的特点。其次，通过实验发现，仅仅选择合适的算法是不够的，参数优化对模型性能的提升起着关键作用。使用GridSearchCV进行参数调优，最终使RMSE从0.9753优化到0.9599，显示出参数调优的重要性。最后，我认为，在实际项目开发中，最为关键的是培养灵活调整模型策略的能力。因为，与学习阶段常用的“完美”数据集不同，现实世界中的数据往往存在各种缺陷和局限。而如何巧妙规避这些数据缺陷，甚至将其转化为模型优化的契机的这种应变能力和解决方案设计能力，不仅是个人技术能力和综合素质的重要体现，也对推动项目成功实施具有决定性作用。

### （二）总结反思

本研究希望构建一个为就餐决策提供解决方案的个性化美食推荐系统，为降低信息成本，我们将用户范围圈定在中船学子，就餐区域限制在定福庄周边，拟设计定福庄周边美食推荐与可视化系统，从而为中传学子就餐选择困难的解决方案。

面对数据获取的局限性，我们创新性地设计了双层推荐架构：第一层基于Surprise框架实现协同过滤算法，从海量数据中精选推荐菜品；第二层通过基于内容的精细化推荐，结合多维度相似度计算确定最终推荐店铺。这种协同过滤与内容推荐相结合的混合策略，不仅有效克服了数据集的局限性，也为餐饮领域的推荐系统开发提供了创新思路。

研究的实践经验表明，构建有效的推荐系统需要在数据处理、算法选择、参数优化等多个环节都做到精益求精。同时，推荐系统的成功不仅取决于算法性能，还需要考虑实际应用场景的特点和需求。这种将理论与实践相结合的经验，为未来类似项目的开发提供了宝贵的参考价值。