推荐系统规范

Recommendation System Standards

1753575刘冬阳

吃啥

目录

[1 程序介绍 2](#_Toc19536327)

[1.1 使用说明 2](#_Toc19536328)

[2 算法详解 3](#_Toc19536329)

[2.1 图像resize算法概述 3](#_Toc19536330)

[2.2 最近邻算法 3](#_Toc19536331)

[2.3 双线性插值 4](#_Toc19536332)

[2.4 双三次插值 5](#_Toc19536333)

[3 测试结果 6](#_Toc19536334)

# 1 推荐系统分析

# 1.1 菜谱推荐系统：

如何处理菜价？该不该将今日菜价作为输入放入神经网络？目前认为不应该。

以DIN为例，这类网络的目的是估计,其中A = {用户喜欢这道菜品}, B ={用户信息；用户兴趣；菜品信息；……}。而事件C = {今日菜价}与事件A和事件B的相关性都不是很大，即用户往往不会因为今天某种蔬菜便宜就变得更喜欢吃用它做的菜肴。因此，和理论上不应该产生很大的差异性。

因此，处于节约计算资源、简化网络架构的考虑，我们认为不应该将今日菜价放入神经网络一起训练，而是作为后处理（post processing）的一部分。

应该先不考虑今日菜价，得到10个左右的高分菜品。然后根据今日菜价制定一些规则，进行筛选。

如何刻画忌口？问题如下：

a.我们可以在前端设置了一个“不喜欢”，点击后可以选择“不喜欢香菜/不喜欢辣的”

b.我们也可以在用户的资料里让他们填写自己的忌口

接下来，我们应该如何对忌口建模。

第一种能想到的办法是将用户的忌口作为一个multihot属性放入神经网络的输入，同样在pooling阶段参与attention机制。优点是对于“一般不吃茄子的人也会喜欢这道茄子做的菜”这种现象能够建模，缺点是相比第二种方法对于忌口的避免可能不够强力。

第二种方法是把忌口也像菜价一样做成一个post-processing。特点是只要用户说了我有这个忌口，最终就肯定不会推荐这东西做的菜，这可能是优点也有可能是缺点

但不论是第一种还是第二种，我们都要专门对“忌口”进行建模，因为通过用户历史行为获取的用户兴趣只能说明用户喜欢什么，不能说明用户不喜欢什么（当然，如果完全不考虑忌口影响也不会太大，因为推荐系统本来就没指望那么准，推荐的大方向对就行，个例出现差错是难免的）。我倾向于使用第二种。

# 2 训练数据请求

2.1 请求流程

1. 由训练机向后端服务器发送请求。

2. 后端服务器向数据库请求对应的历史点击数据，并将这些数据按照规定的格式存储在csv文件中。

3. 将该csv文件发送给训练机。

4. 训练集接收并保存该文件，基于该文件进行训练。

2.2 请求形式

|  |  |
| --- | --- |
| 根据时间范围获取用户点击记录 | |
| method | GET |
| url | api/recommendation/trainTimeSpan |
| data | {  start\_time:  end\_time:  } |
|  |  |
|  |  |
|  |  |
|  |  |

2.3 输入数据列表

|  |  |  |  |
| --- | --- | --- | --- |
| 用户域 | | | |
| 信息名称 | 信息描述 | 信息取值 | 信息取值 |
| gender | 用户性别 | 0,1 | 0,1 |
| age | 用户年龄段 | 0,1,2,3,4 | 0,1,2,3,4 |
| born\_place | 用户籍贯 | 每个省份对应一个编码，onehot | 23 |
| preference | 用户偏好 | 每个口味代表一个编码，multihot | 1|4|5 |
| recipe\_records | 前20条浏览记录的菜品id | 每个菜品的id，multihot（历史不足时可以少于20条） | 201|302|1|204|3|2|201|5|100 |
| 菜谱域 | | | |
| recipe\_id | 菜谱id | 菜谱的id | 1 |
| recipe\_taste | 菜品味道 | 每个口味代表一个编码，与preference属性一致，multihot | 2|4|5 |
| recipe\_style | 菜品菜系 | 每个已知菜系代表一个编码 | 3|6|8 |
| ingredient | 原材料 | 原材料列表，multihot，为多个原材料id的集合 | 1|2|5|100 |
| seasoning | 调料 | 调料列表，为多个调料id的集合 | 2|5|10 |

2.4 csv文件细节要求

1. Id类型不要在前面补0，如西红柿炒鸡蛋的id要写成1而不是00000001。
2. 用户的preference和菜品的recipe\_taste要有一致的编码，即refer to同一个taste表
3. 当用户的历史recipe\_records不足20条时，只需将全部记录输出，不需要引入其他的占位或填充
4. 未知属性用NA占位

# 3 预测数据请求

3.1 请求流程

1. 前端向业务服务器请求获取菜谱推荐
2. 业务服务器向预测服务器发出请求，并呈递用户相关信息
3. 预测服务器接收请求，对数据进行处理并给出推荐系统输出结果
4. 预测服务器将推荐系统的输出返回给业务服务器
5. 业务服务器将结果返回给前端

3.2 输入数据列表

|  |  |  |  |
| --- | --- | --- | --- |
| 用户域 | | | |
| 信息名称 | 信息描述 | 信息取值 | 信息取值 |
| gender | 用户性别 | 0,1 | 0,1 |
| age | 用户年龄段 | 0,1,2,3,4 | 0,1,2,3,4 |
| born\_place | 用户籍贯 | 每个省份对应一个编码，onehot | 23 |
| preference | 用户偏好 | 每个口味代表一个编码，multihot | 1|4|5 |
| recipe\_records | 前20条浏览记录的菜品id | 每个菜品的id，multihot（历史不足时可以少于20条） | 201|302|1|204|3|2|201|5|100 |

# 4 预测服务器对菜谱域信息定期索取

4.1 请求流程

1. 预测服务器向业务服务器请求获取实时菜谱域信息

2. 菜谱域信息从业务服务器以csv文件的格式发送给预测服务器

3. 预测服务器接收并保存csv文件中的数据

4.2 请求形式

|  |  |
| --- | --- |
| 根据时间范围获取用户点击记录 | |
| method | GET |
| url | api/recommendation/trainTimeSpan |
| data | {  start\_time:  end\_time:  } |
|  |  |

4.3 输入数据列表

|  |  |  |  |
| --- | --- | --- | --- |
| 菜谱域 | | | |
| recipe\_id | 菜谱id | 菜谱的id | 1 |
| recipe\_taste | 菜品味道 | 每个口味代表一个编码，与preference属性一致，multihot | 2|4|5 |
| ingredient | 原材料 | 原材料列表，multihot，为多个原材料id的集合 | 1|2|5|100 |
| seasoning | 调料 | 调料列表，为多个调料id的集合 | 2|5|10 |
| 菜价域 | | | |
| ingredient\_id | 食材id | 食材的id | 1 |
| price | 食材当前价格 | 一个浮点数，单位为元/斤 | 3.5 |
| price\_history | 价格历史 | ？ |  |