深圳大学考试答题纸

(以论文、报告等形式考核专用)

二○二一～二○二二学年度第一学期

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程编号 | 1502950001 | | 课程名称 | | 数据挖掘导论 | | 主讲教师 | | 陈小军 | 评分 |  |
| 学 号 | 2019092121 | | 姓名 | 沈晨玙 | | 专业年级 | | 19级计算机科学与技术国际班 | | | |
|  | | | | | | | | | | | |
| 教师评语： | | | | | | | | | | | |
| 项目名称： | | 推荐系统 - 新闻推荐 | | | | | | | |  | |

摘 要：

完成了一整套的新闻推荐系统。根据用户过往浏览记录，为每一位用户个性化的推荐合适的新闻，并进行了评估测试。

本次系统包含了五大阶段：数据收集阶段，召回阶段，多路召回阶段，排序阶段，预测阶段。本报告详细的介绍了每一阶段之中的原理以及实现细节，包括自己的创新点。

最终在天池的长期赛中，达到了17 / 7712、TOP 0.22%的成绩。

**深圳大学课程项目报告**

|  |
| --- |
| **课程名称： 数据挖掘导论**  **项目名称： 推荐系统 - 新闻推荐**  **学 院： 计算机与软件学院**  **专 业： 计算机科学与技术**  **任课教师： 陈小军**  **报 告 人： 沈晨玙 学号： 2019092121**  **提交时间： 2021年12月16日星期四** |

**教 务 处 制**

**一、项目知识点**

1. 数据预处理；
2. 数据可视化；
3. 特征工程；
4. 模型选择；
5. 实验结果的评价；

**二、实验过程**

1. 比赛任务分析
   1. **赛题背景**

赛题以新闻APP中的新闻推荐为背景，要求选手根据用户历史浏览点击新闻文章的数据信息预测用户未来点击行为，即用户的最后一次点击的新闻文章

* 1. **赛题数据**

数据来自某新闻APP平台的用户交互数据，包括30万用户，近300万次点击，共36万多篇不同的新闻文章，同时每篇新闻文章有对应的embedding向量表示。将会从中抽取20万用户的点击日志数据作为训练集，5万用户的点击日志数据作为测试集A，5万用户的点击日志数据作为测试集B。

|  |  |
| --- | --- |
| train\_click\_log.csv | 训练集用户点击日志 |
| testA\_click\_log.csv | 测试集用户点击日志 |
| articles.csv | 新闻文章信息数据表 |
| articles\_emb.csv | 新闻文章embedding向量表示 |
| sample\_submit.csv | 提交样例文件 |

数据表

字段表

|  |  |
| --- | --- |
| ****Field**** | ****Description**** |
| user\_id | 用户id |
| click\_article\_id | 点击文章id |
| click\_timestamp | 点击时间戳 |
| click\_environment | 点击环境 |
| click\_deviceGroup | 点击设备组 |
| click\_os | 点击操作系统 |
| click\_country | 点击城市 |
| click\_region | 点击地区 |
| click\_referrer\_type | 点击来源类型 |
| article\_id | 文章id，与click\_article\_id相对应 |
| category\_id | 文章类型id |
| created\_at\_ts | 文章创建时间戳 |
| words\_count | 文章字数 |
| emb\_1,emb\_2,…,emb\_249 | 文章embedding向量表示 |

* 1. **评价指标**

利用推荐系统常用的两个指标MRR与HR进行评估。

HR(Hit Rate):

命中率：预测个数占用户总数的比例 （HR\_i表明前i篇文章的HR得分）

MRR(Mean Reciprocal Rank)：

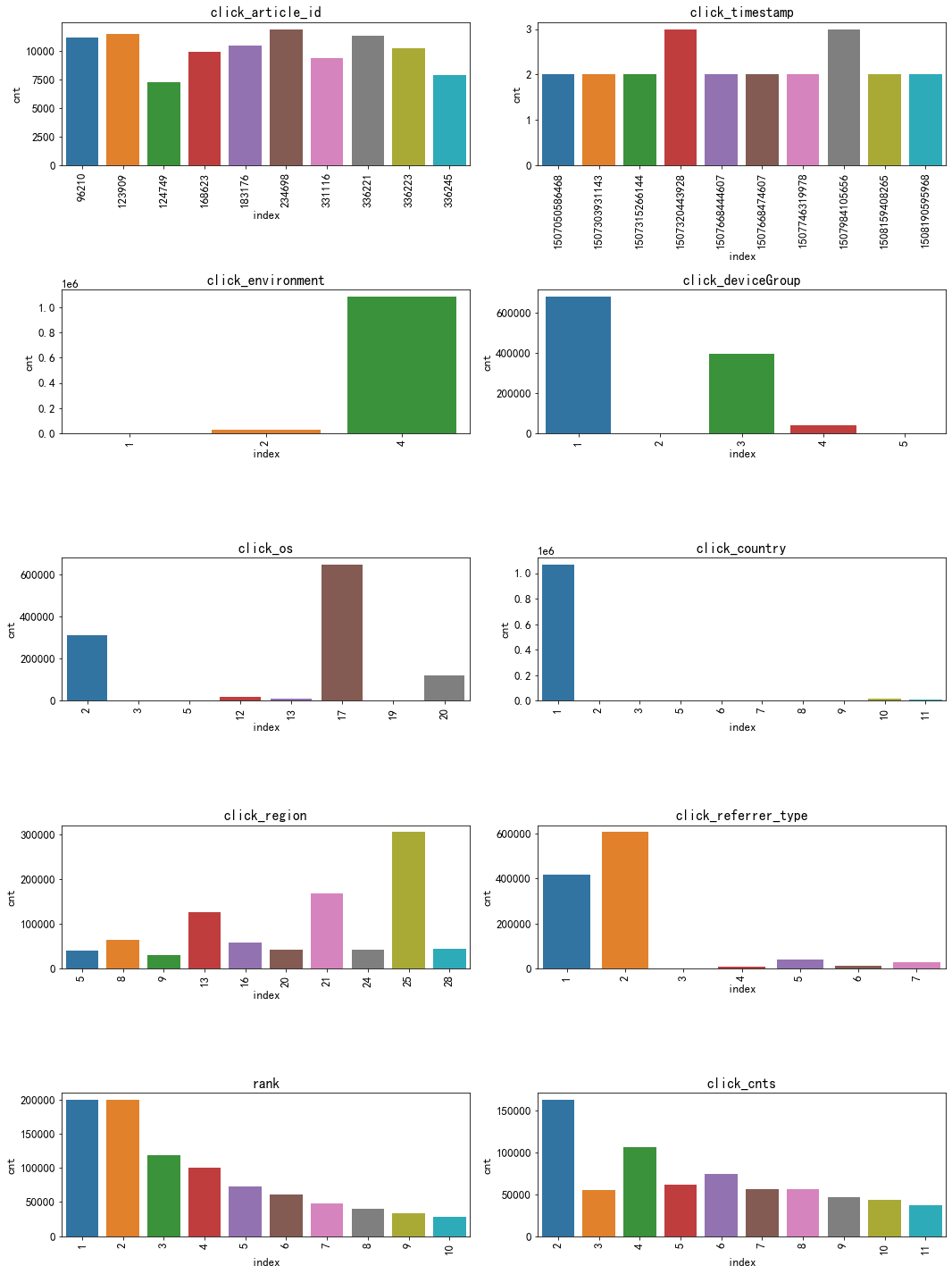
首先对选手提交的表格中的每个用户计算用户得分

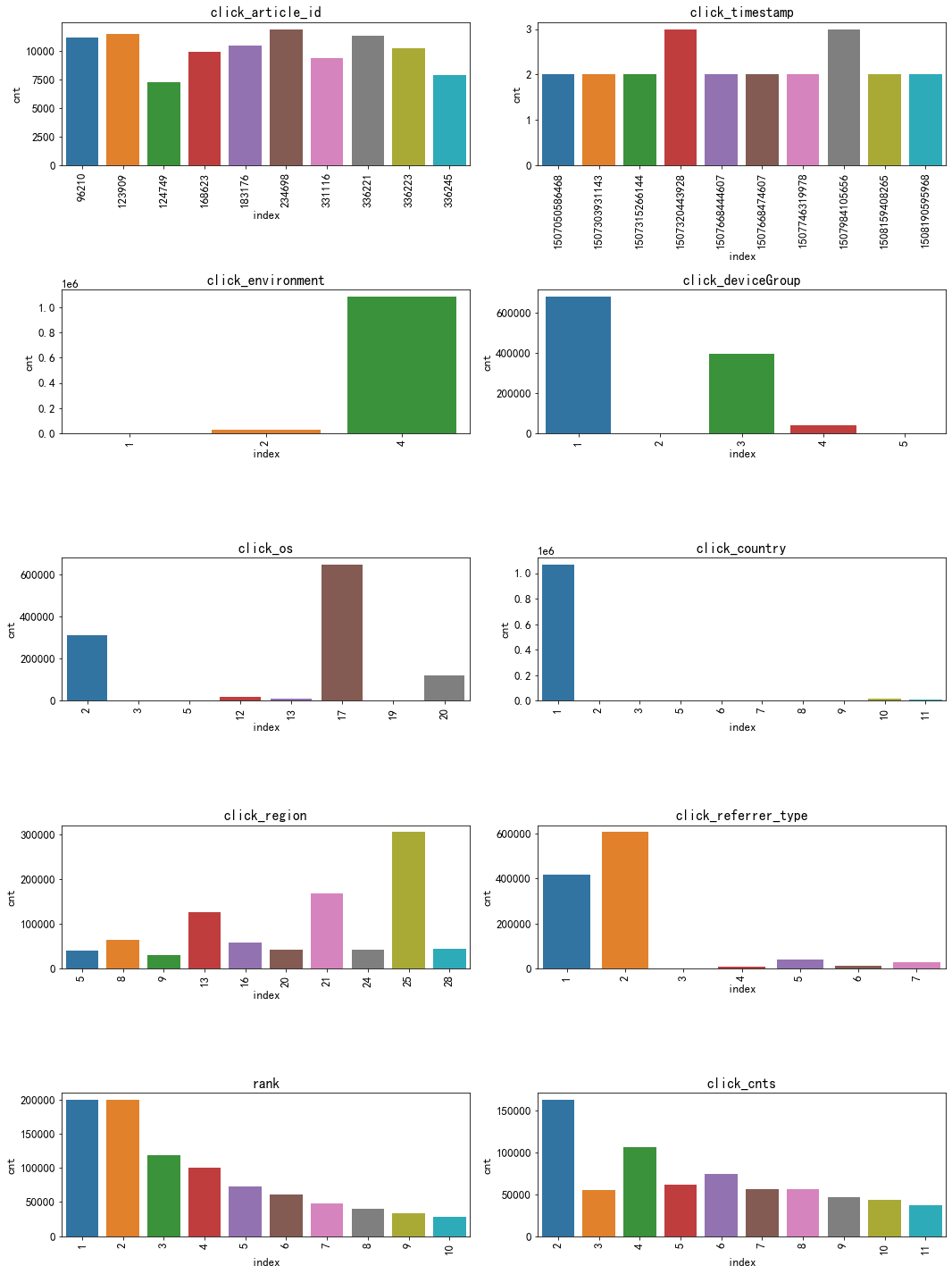
其中, 如果选手对该user的预测结果predict k命中该user的最后一条购买数据则s(user,k)=1; 否则s(user,k)=0。而选手得分为所有这些score(user)的平均值。（MRR\_i表明前i篇文章的MRR得分）

* 1. **赛题难点**
     1. 数据量大：一共有包括30万用户，近300万次点击，共36万多篇不同的新闻文章
     2. 推荐系统随机性大：由于是基于真实数据，每个用户的点击随机性较高
  2. **赛题分类**

最终需要给用户从高到低推荐5篇文章进行预测，因此为排序问题，可以通过一定方式转换为二分类问题（后续会提及方法）。

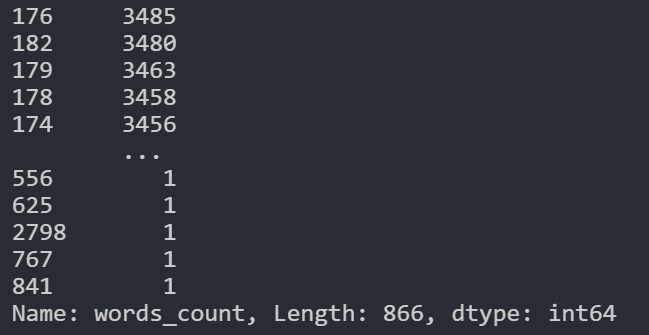
1. 数据统计与可视化分析
   1. **数据集可视化展示**
      1. Click\_log（用户点击记录）





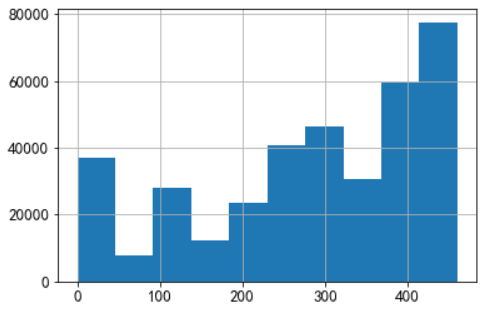
* + 1. Articles.csv（文章信息）

统计单词数量

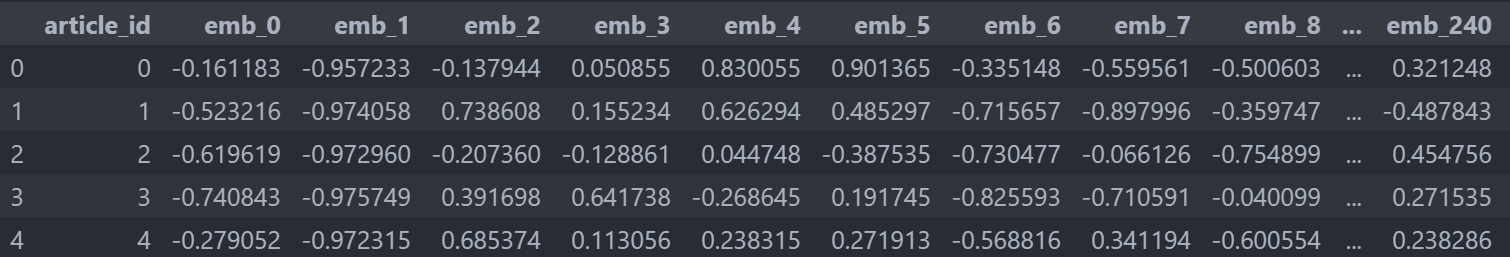


统计文章主题

共461个主题，分布如下



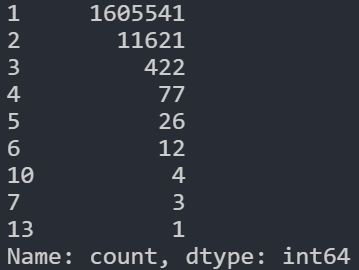
* + 1. Article\_emb.csv（文章向量）



* 1. **数据分析**
     1. 用户特征

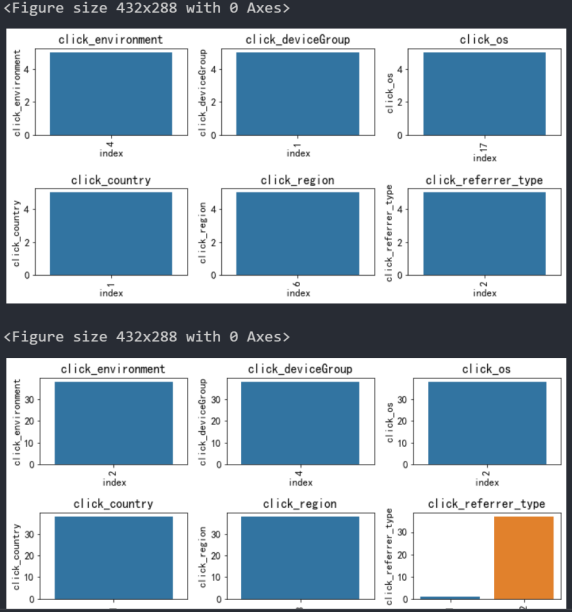
用户重复点击次数

有1605541（约占99.2%）的用户未重复阅读过文章，仅有极少数用户重复点击过某篇文章。



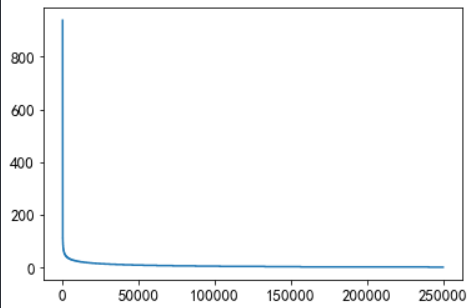
用户点击环境变化

随机采样多名用户的点击环境变化，可以看到大部分用户的点击环境是相对固定的。



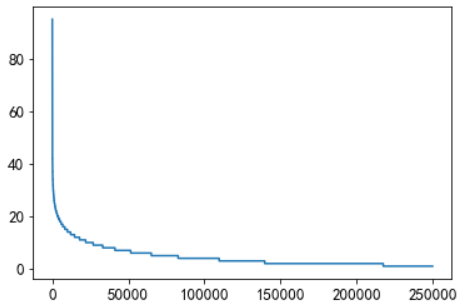
用户点击新闻总次数

可以看到大部分用户点击次数均在50次以下，可以以此为活跃/不活跃用户的分界值。



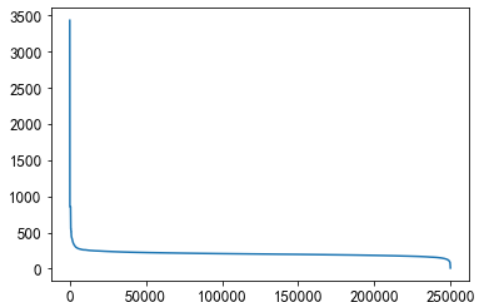
用户点击新闻类型统计

一小部分用户阅读类型是极其广泛的，大部分人都处在20个新闻类型以下。



用户查看新闻字数长度统计

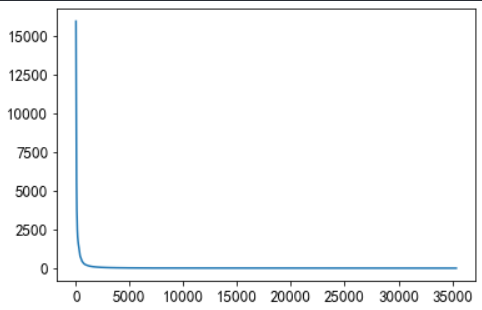
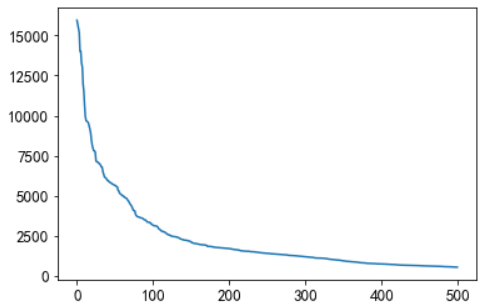
有一小部分人看的文章平均词数非常高，也有一小部分人看的平均文章次数非常低。大多数人偏好于阅读字数在200-400字之间的新闻。大多数人都是看250字以下的文章。



* + 1. 文章特征

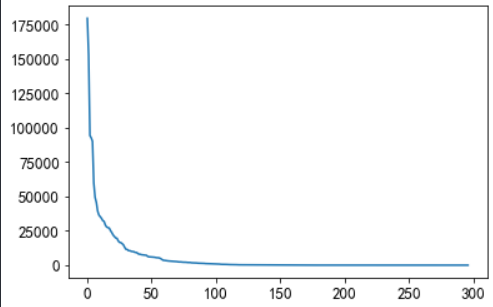
新闻被点击总次数

可以看到大部分新闻被点击总次数在1000次以下，可以以此为热门/不热门新闻的分界值。

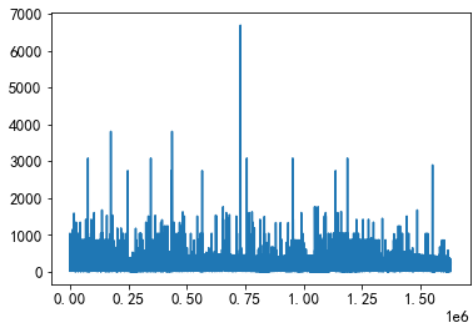
 

新闻具体特征

不同类型的新闻出现的次数

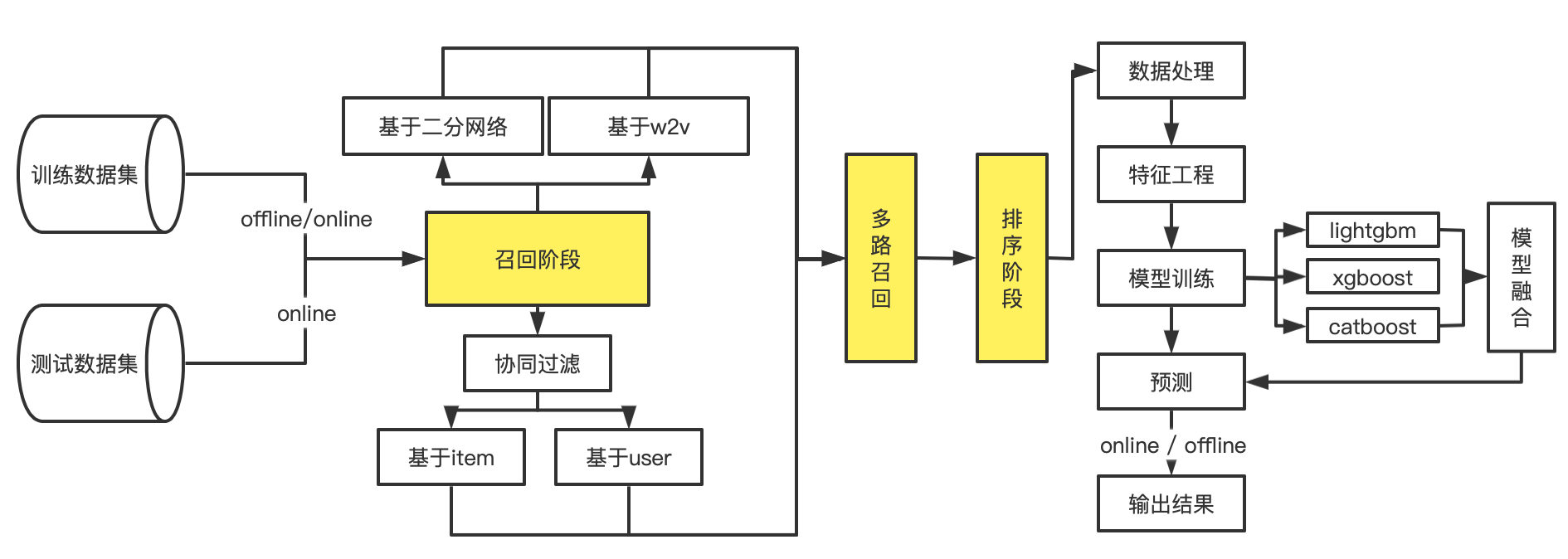


文章字数的统计



1. 推荐系统基础架构综述

推荐系统是一个比较复杂的架构，本次比赛我所使用的具体架构如下图所示：



推荐系统整体分为五大阶段：数据收集阶段，召回阶段，多路召回阶段，排序阶段，预测阶段。

后续四个大题将会更细节的展示每个阶段其中的原理作用及实现细节。

1. 数据收集阶段

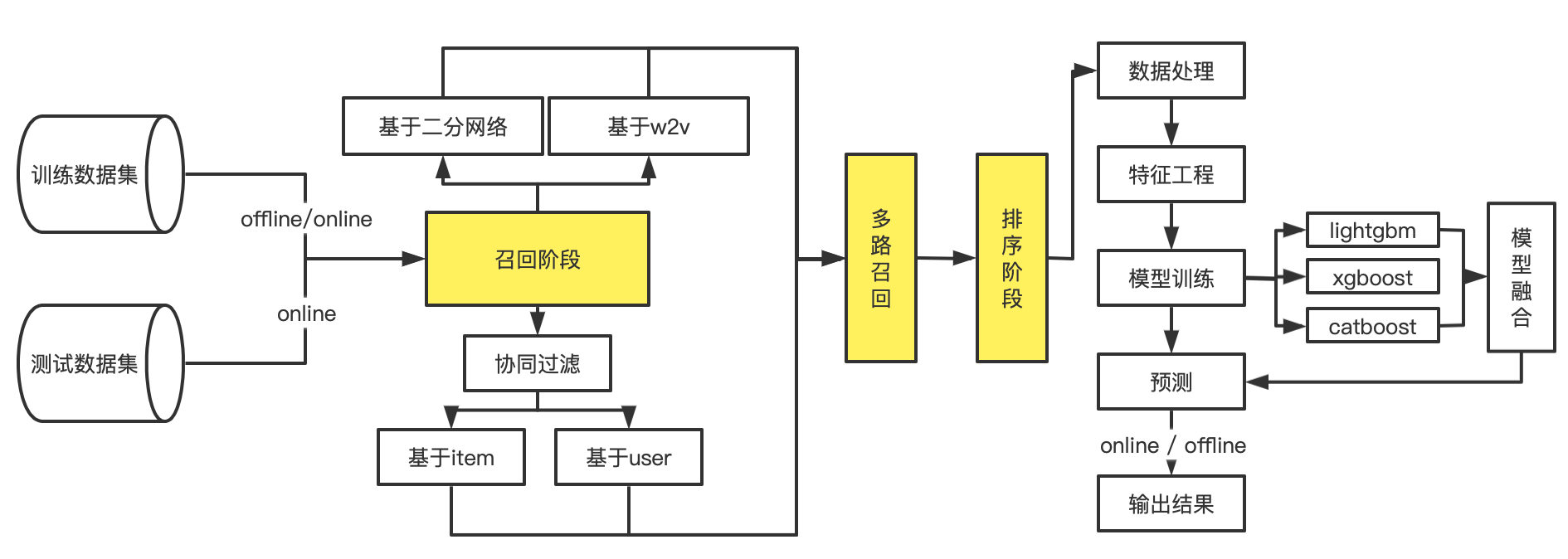
推荐系统的最终目的是完成对于线上用户的实时推荐任务，所以分为了离线/在线两种模式（offline/online）。

两种模式的区别：

离线模式是为了在系统上线前进行测试用的，所以需要将整体的数据集分为训练数据集以及测试数据集，即训练数据集来自trainData，并且通过testData对模型进行评估。

在线模式是为了完成线上用户的实时推荐任务，所以可以将全部的数据均用于训练以提升数据丰富性，即训练数据集来自trainData+testData，并且因为需要部署在线上，所以不需要进行线下模型评估。

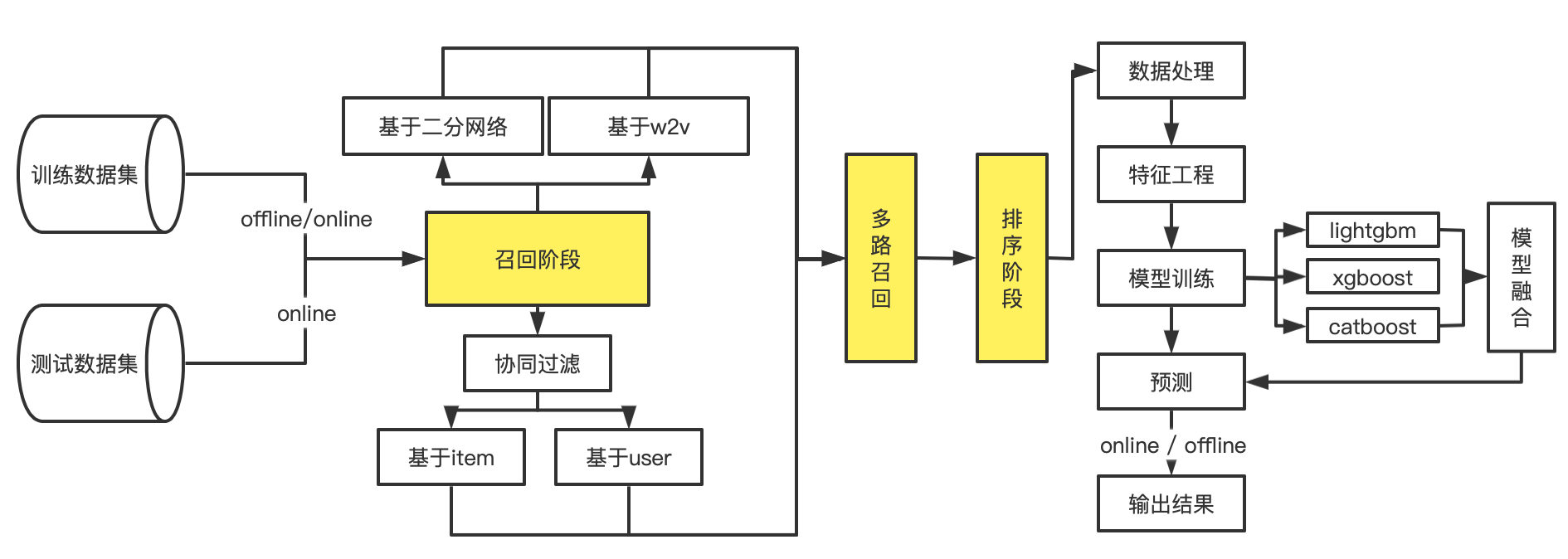
在下图中可以清晰的看到不同模式下的数据来源：



1. 召回阶段

收集完数据后的下一步便是对新闻列表进行第一步粗选，即进入召回阶段。

召回阶段整体架构图：



召回阶段意义：在海量的新闻列表（36万多篇）之中，有针对性的对于每一位用户挑选出一部分新闻进行推荐（通常在100-300篇之间）。

本次比赛中，我一共使用了4中召回的方法。

1. 协同过滤算法

基于物品的协同过滤（itemCF）

基于用户的协同过滤（userCF）

2. 基于二分网络的召回方法

3. 基于word2vec的召回方法

接下来将一一进行介绍

1. 基于物品的协同过滤（itemCF）
   1. 整体思路：给用户推荐与曾经读过文章相似的文章
   2. 步骤：
      1. 计算物品之间的相似度

根据物品历史被喜欢的情况，假如某两个物品历史共同被许多用户喜欢，则说明这两个物品是相似的。

假设喜欢物品a的用户数为N(a)，喜欢物品b的用户数为N(b)，那么a与b的相似度为：

上述公式可以理解为喜欢 A 物品的用户中，有多少比例的用户也喜欢B，比例越高，说明A与 B 的相似度越高。

但是这样的公式有一个问题，如果物品 B 很热门，很多人都喜欢，那么相似度就会无限接近1，这样就会造成所有的物品拿出来，都与 B 有极高的相似度，这样就没有办法证明物品之间的相似度是可靠的了。

为了避免出现类似的情况，可以通过以下公式进行改进：

**\*核心代码：2\_recall\_itemCF.py 中cal\_sum 函数**

* + 1. 根据物品的相似度和用户的历史行为计算推荐分

获得了物品的相似度后，则根据以下公式来计算用户 u 对物品 b 的兴趣：

其中，N(u)是用户喜欢物品的集合，S(b,K)是和物品b最相似的K个物品的集合，Wab是物品a和b的相似度，Rua是用户u对物品a的兴趣。

**\*核心代码：2\_recall\_itemCF.py 中recall 函数**

* + 1. 根据推荐Top100的文章给用户

第二步中最近计算出许多文章的推荐分，根据从高到低的原则，召回得分最高的前100篇文章给每一位用户。

* 1. 创新点：
     1. 引入文章关联机制

因为本次比赛数据量巨大，有多达36万篇的文章，所以不可能对所有文章进行两两相似度计算，由此引入文章关联机制，方法如下：

假设A用户读过X文章，则对于所有读过X文章的所有用户，X与其读过的其他文章均为关联文章。例如：A读过X，B读过X、Q、W、E、R，则对于XQWER为关联文章。

如此操作，可以大大减少计算量，将本问题变得可解。根据上述策略得到的推荐文章，HR\_5 = 0.29678，MRR\_5 = 0.16752

* + 1. 引入新颖度概念

由于文章的流行度分布呈长尾分布，所以为了流行度的平均值更加稳定，在计算平均流行度时对每个物品的流行度取对数。

**\*核心代码**

1. sim\_dict[item][relate\_item] += loc\_weight / \
2. math.log(1 + len(items))

此时效果得到提升：HR\_5 = 0.30320，MRR\_5 = 0.17134

* + 1. 考虑文章点击顺序

根据每一位用户的点击历史，点击顺序也会对文章相似度计算产生影响。

例如：

对于点击顺序ABC与点击顺序ACB，一般而言用户倾向于首先点击自己的更喜欢的，因此B的推荐优先级大于C。

对于点击顺序AB与点击顺序A......B，显然虽然同样是关联文章，但是前者的相似性要远远大于后者。

**\*核心代码**

1. # 可调参数
2. loc\_alpha = 1.0 **if** loc2 > loc1 **else** 0.7
3. loc\_weight = loc\_alpha \* (0.9\*\*(np.abs(loc2 - loc1) - 1))
5. sim\_dict[item][relate\_item] += loc\_weight / \
6. math.log(1 + len(items))
7. **for** loc, item **in** enumerate(interacted\_items):
8. **for** relate\_item, wij **in** sorted(item\_sim[item].items(), key=**lambda** d: d[1], reverse=True)[0:200]:
9. **if** relate\_item **not** **in** interacted\_items:
10. rank.setdefault(relate\_item, 0)

            rank[relate\_item] += wij \* (0.7\*\*loc)

* + 1. 召回时引入位置距离衰减

新闻点击是强热点相关，所以历史点击新闻对下一次点击预测的影响传播不会太远。在实际测试中，利用所有历史点击新闻做召回，hitrate\_5 指标只有0.31701，限定只用最近点击的两个新闻来做召回的话，可以提升至0.32958。

|  |  |  |
| --- | --- | --- |
| **考虑文章数** | **HR\_5** | **MRR\_5** |
| ALL | 0.31701 | 0.18735 |
| 3 | 0.32125 | 0.18858 |
| 2 | 0.32958 | 0.19303 |
| 1 | 0.19370 | 0.11005 |

**\*核心代码**

1. interacted\_items = interacted\_items[::-1][:2]
   1. 最终效果：

HR\_5 = 0.32958，MRR\_5 = 0.19303

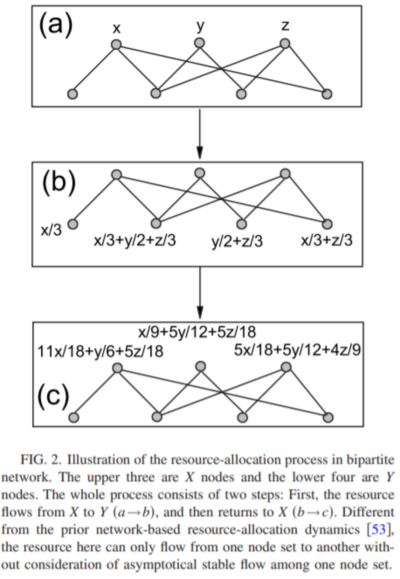
1. 基于用户的协同过滤（userCF）
   1. 整体思路：给用户推荐与其相似的其他读者曾经读过的文章
   2. 步骤：
      1. 计算用户之间的相似度
      2. 根据用户的相似度和用户的历史行为给用户推荐
      3. 根据推荐Top100的文章给用户
   3. 遇到困难：
      1. 用户量数据量太大，有30W的用户，因此需要一个30W\*30W的矩阵用于计算用户相似性。
      2. 计算时间太长，且256G内存也不够

因此此方案最终没能实现，被否决了。

1. 基于二分网络的召回方法

注：此方法来自论文《How to project a bipartite network?》https://arxiv.org/abs/0707.0540

* 1. 整体思路：与基于物品的协同过滤思路类似，区别在于相似性矩阵中的细节。
  2. 步骤：
     1. 计算物品之间的相似度
     2. 根据物品的相似度和用户的历史行为给用户推荐
     3. 根据推荐Top100的文章给用户创新点：
  3. 创新点：
     1. 原论文创新点：
        1. 两个新闻的共同被点击用户过多，则相似度减少
        2. 共同被点击用户的点击新闻过多，相似度也要减少。
        3. 核心思想如下图所示



* + 1. 召回时引入位置距离衰减

|  |  |  |
| --- | --- | --- |
| **考虑文章数** | **HR\_5** | **MRR\_5** |
| ALL | 0.19370 | 0.11005 |
| 3 | 0.27720 | 0.15527 |
| 2 | 0.31212 | 0.17854 |
| 1 | 0.33597 | 0.20273 |

新闻点击是强热点相关，所以历史点击新闻对下一次点击预测的影响传播不会太远。在实际测试中，利用所有历史点击新闻做召回，hitrate\_5 指标只有0.19370，限定只用最近点击的一个新闻来做召回的话，可以大幅提升至0.33597。

**\*核心代码**

1. interacted\_items = interacted\_items[::-1][:1]
   * 1. 移除新颖度的概念

理论上若引入会达到更好的效果，但是实验测试下来，不如移除。

移除前：HR\_5 = 0.33597 MRR\_5 = 0.20273

移除后：HR\_5 = 0.35613 MRR\_5 = 0.20928

**\*核心代码**

1. **for** user **in** users:
2. tmp\_len = len(user\_item\_dict[user])
3. **for** relate\_item **in** user\_item\_dict[user]:
4. sim\_dict[item].setdefault(relate\_item, 0)
5. sim\_dict[item][relate\_item] += 1 / \
6. ((len(users)+1) \* (tmp\_len+1))
   1. 最终效果：

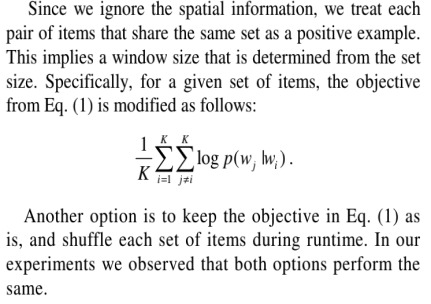
HR\_5 = 0.35613，MRR\_5 = 0.20928

1. 基于word2vec的召回方法

注：此方法来自论文《ITEM2VEC: NEURAL ITEM EMBEDDING FOR COLLABORATIVE FILTERING》https://arxiv.org/abs/1603.04259

* 1. 整体思路：使用序列学习模型Word2Vec学习文章向量，从而推荐相似的文章
  2. 步骤：
     1. 将文章列表通过Word2Vec转化为256维的向量。
     2. 根据文章向量计算文章相似度。
     3. 根据相似度计算推荐Top100的文章给用户
  3. 论文核心思想：

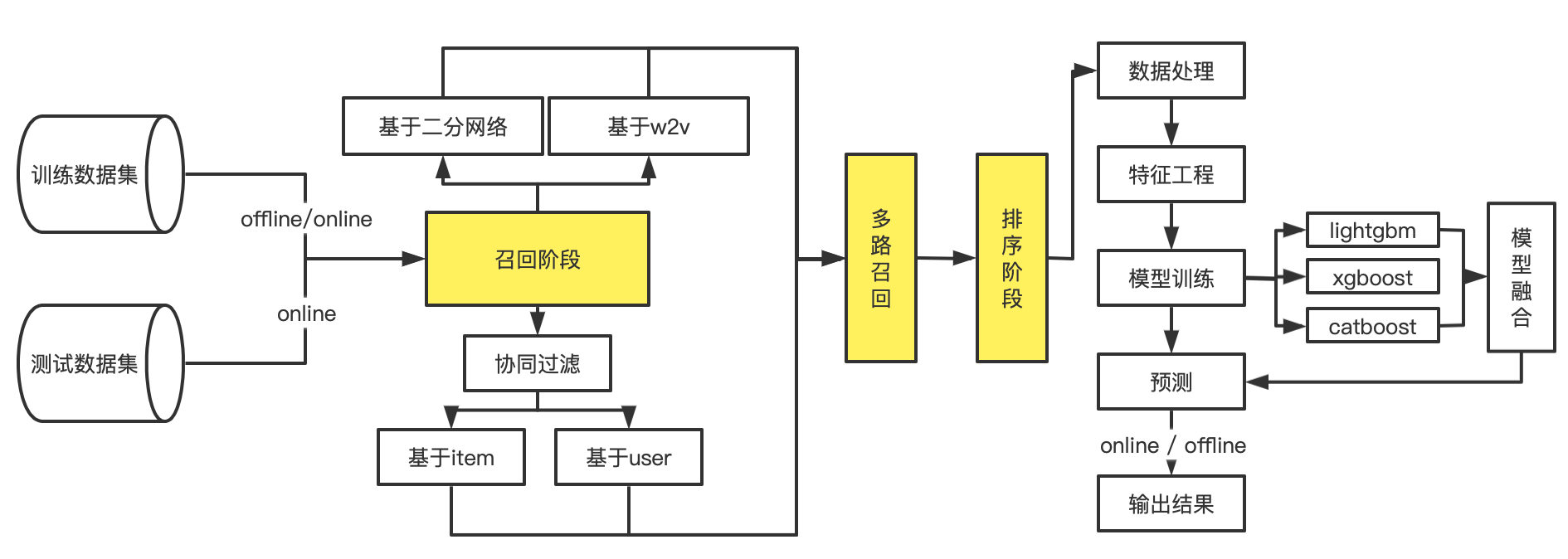
任何物品都可以转换为向量，即item2vec的思想。



1. 多路召回阶段

召回阶段完成后会返回多组召回结果，下一步就是整合思路召回结果，进入多路召回阶段。

流程图：



多路召回结果之间肯定存在重复召回的情况，这也是为什么召回策略讲究差异性，其实就是为了减少重复召回的数量。

多路召回核心：重复召回的文章在不同召回策略中的得分是不同的，需要对得分进行合并处理。按照一定比例计算每篇文章的得分，并重新召回，得到多路召回的召回结果。

方法：

1. 按照一定比例进行融合

itemCF：二分网络：w2v = 2 : 2.5 :0.1

**\*核心代码**

1. weights = {'itemcf': 1, 'binetwork': 1, 'w2v': 0.1}
2. recall\_result['sim\_score'] = recall\_result['sim\_score'] \* weight
3. # 合并召回结果
4. recall\_final = pd.concat(recall\_list, sort=False)
5. recall\_score = recall\_final[['user\_id', 'article\_id','sim\_score']]groupby(['user\_id', 'article\_id'])['sim\_score'].sum().reset\_index()

其中在合并召回的时候测试了sum，mean和max，效果对比见下表。max丢失的消息较多，mean对重复次数多的新闻不公平。可以看到对于召回前提升效果明显。

HR\_5 = 0.47988，MRR\_5 = 0.27815

|  |  |  |
| --- | --- | --- |
| **合并策略** | **HR\_5** | **MRR\_5** |
| sum | 0.47988 | 0.27815 |
| mean | 0.42551 | 0.23820 |
| max | 0.45905 | 0.26015 |

1. 删除没有召回到真实点击的验证集用户，减少了无用负样本的数量。

**\*核心代码**

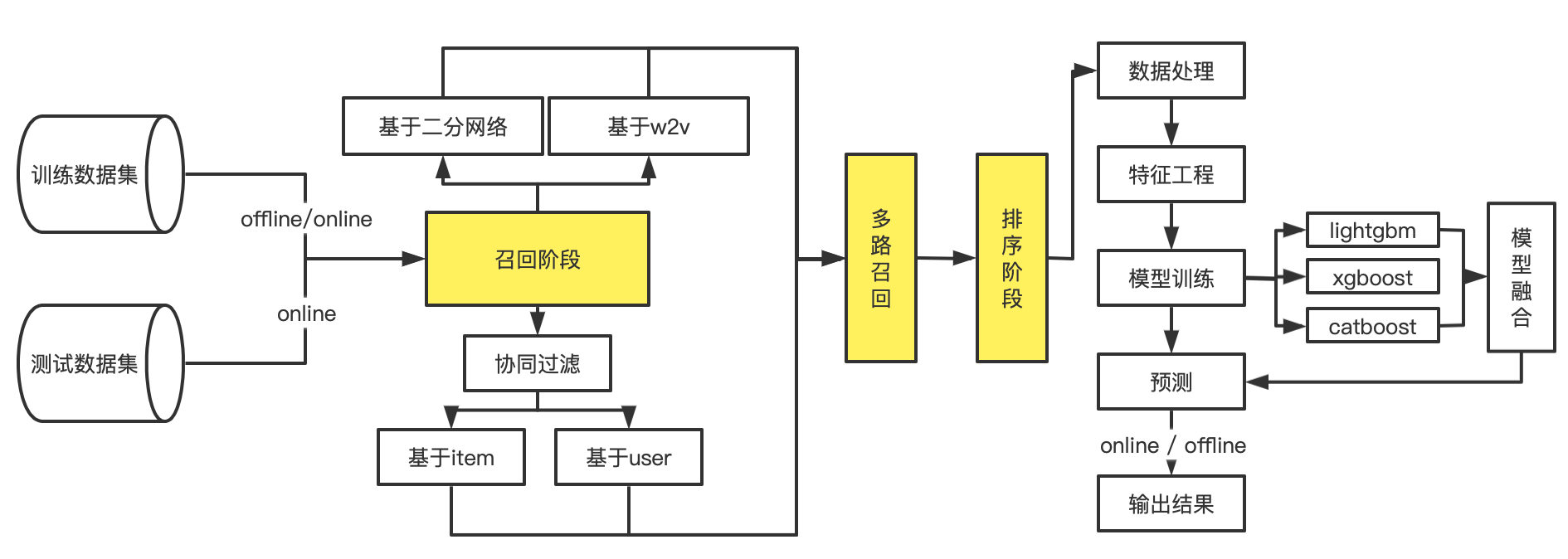
1. # 删除无正样本的训练集用户
2. gg = recall\_final.groupby(['user\_id'])
3. useful\_recall = []
5. **for** user\_id, g **in** tqdm(gg):
6. **if** g['label'].isnull().sum() > 0:
7. useful\_recall.append(g)
8. **else**:
9. label\_sum = g['label'].sum()
10. **if** label\_sum > 1:
11. **print**('error', user\_id)
12. **elif** label\_sum == 1:
13. useful\_recall.append(g)

最终每个用户平均召回151篇文章。

1. 排序阶段

完成了多路召回阶段后就是对于召回的文章进行排序，进入排序阶段。其中包括数据处理、特征工程、模型训练、模型融合等部分。

排序阶段整体架构图：



排序阶段的意义：在召回阶段召回的文章之中，针对性每一位用户的所有文章进行排序，选择出最终的推荐文章（精选）。

排序阶段主要分为两个部分：数据处理/特征工程 以及 模型训练/模型融合。

接下来将进行一一介绍。

1. 数据处理 / 特征工程

数据集给出的特征信息较少，所以数据处理及特征工程部分主要围绕交互属性展开。

故添加一下字符段用于特征工程：

* + 1. 新闻特征
* 新闻字数
* 新闻创建时间
* 新闻被阅读数量
  + 1. 用户特征
* 用户点击新闻的创建时间差的平均值
* 用户点击新闻的点击时间差的平均值
* 用户点击新闻的点击-创建时间差的统计值：mean，std
* 用户点击新闻的点击时长统计值
* 用户点击新闻的字数统计值
* 用户点击新闻的创建时间统计值
* 用户点击新闻的点击时间统计值
* 用户新闻阅读数量
* 用户某种类新闻阅读数量
  + 1. 用户-新闻交互特征
* 待预测新闻和用户所有历史点击新闻相似度按次序加权求和
* 待预测新闻和用户最近一次点击新闻相似度

**\*核心代码：6\_rank\_feature.py**

1. 模型训练 / 模型融合

我们的训练数据是29个特征+一个标签（1/0代表点击/未点击），需要完成的任务是对于多路召回阶段返回文章进行点击概率排序任务。

所以可以将排序问题转换为二分类问题（点击/未点击），根据文章的点击概率进行排序。

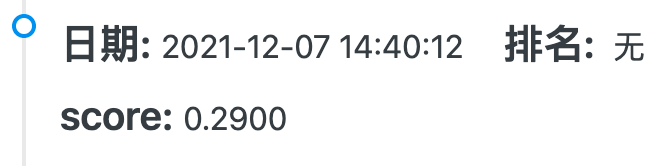
本次比赛一共尝试了三种分类模型用于预测：LGBMClassifier、b)XGBClassifier、CatBoostClassifier，并统计了每种模型下的特征重要性指数。

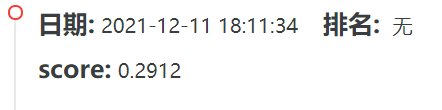
* + - * 1. LGBMClassifier

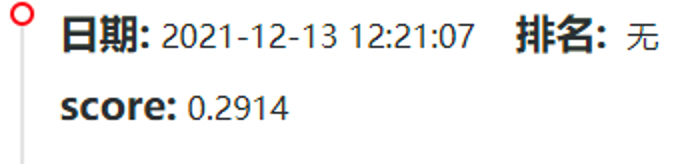
经测试，已知最佳模型参数：

1. model = lgb.LGBMClassifier(num\_leaves=80,
2. max\_depth=9,
3. learning\_rate=0.01,
4. n\_estimators=100000,
5. subsample=0.8,
6. feature\_fraction=0.8,
7. reg\_alpha=0.5,
8. reg\_lambda=0.5,
9. random\_state=seed,
10. importance\_type='gain',
11. metric=None)

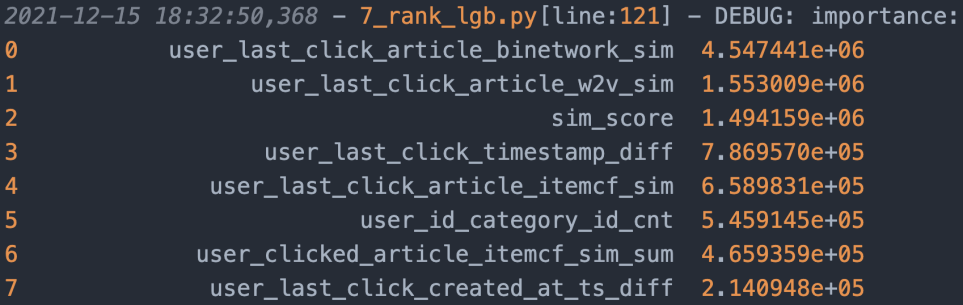
**\*核心代码：7\_rank\_lgb.py**

部分成绩展示：





特征重要性指数排序：可以看出，未来点击文章与最后一次的点击文章高度相关

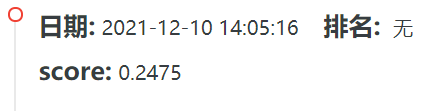


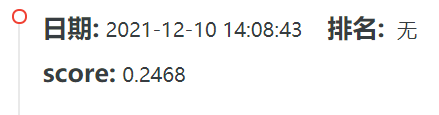
* + - * 1. XGBClassifier

经测试，已知最佳模型参数：

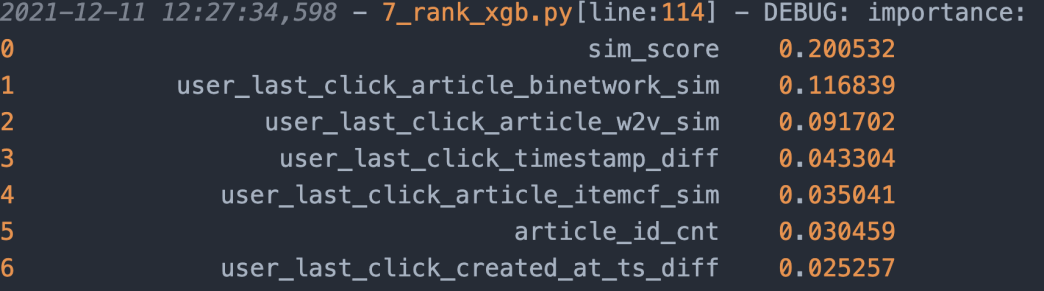
1. model = xgb.XGBClassifier(max\_depth=10,
2. learning\_rate=0.08,
3. num\_leaves=64,
4. n\_estimators=1000000000,
5. subsample=0.8,
6. feature\_fraction=0.75,
7. reg\_alpha=0.7,
8. reg\_lambda=1.2,
9. random\_state=seed,
10. eval\_metric='AUC',
11. tree\_method='gpu\_hist')

**\*核心代码：7\_rank\_xgb.py**

部分成绩展示：



特征重要性指数排序：可以看出，未来点击文章与最后一次的点击文章高度相关

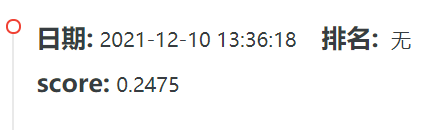


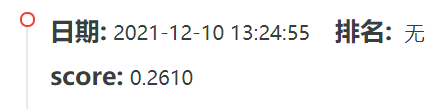
* + - * 1. CatBoostClassifier

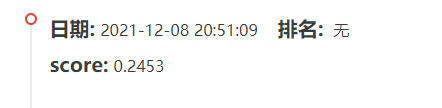
经测试，已知最佳模型参数：

1. model = cbt.CatBoostClassifier(max\_depth=16,
2. learning\_rate=0.08,
3. n\_estimators=10000,
4. subsample=0.75,
5. random\_state=seed,
6. eval\_metric='AUC',
7. bootstrap\_type='Poisson',
8. task\_type='GPU')

**\*核心代码：7\_rank\_cbt.py**

部分成绩展示：



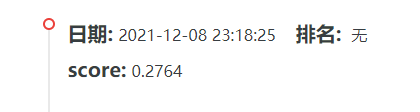


特征重要性指数排序：可以看出，未来点击文章与最后一次的点击文章高度相关

测试完了三种单模型预测，我尝试进行了三种模型的融合。

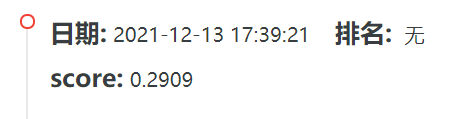
方案一：将三种模型分别得到的最高得分结果进行整合合并。将共同推荐的文章放置在最前，其余位置按照三者之中最高分模型剩余文章进行填充。（平均召回重复率3.3篇）

最终融合得分：



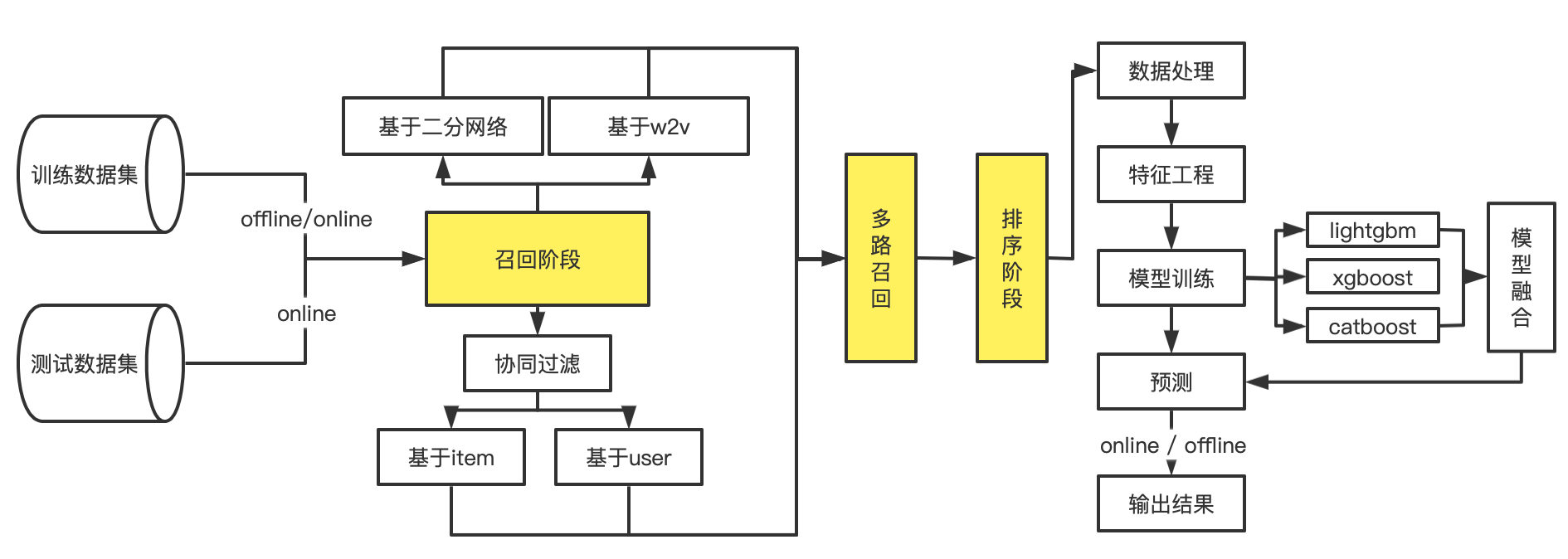
方案二：将所有模型中得分最高的三份结果进行整合合并。将共同推荐的文章放置在最前，其余位置按照三者之中最高分模型剩余文章进行填充。（由于LGBM模型得分远远高于其他两种模型，因此三者均为LGBMClassifier）

最终融合得分：



很可惜两种方法都没能将比赛得分进一步提高，均不如单一模型得分高。

1. 预测阶段



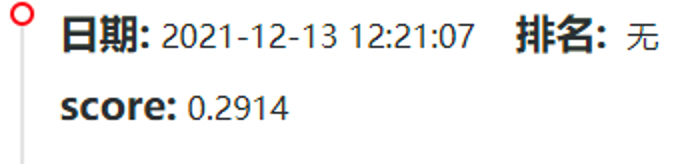
完成了模型训练及评估后，进行最后的预测阶段。

将模式调整为online模式，利用已经训练完的模型进行结果提交，完成比赛。

**三、项目结果**

最终分数：0.2914

排名：17 / 7712 top 0.22%







**四、项目总结**

通过这次比赛了解了推荐系统中的一些基本算法，在数据挖掘领域有了更深的了解。

在进行数据处理之前，首先需要进行数据可视化操作。在这一阶段，将表格式的大数据进行可视化，可以更加清晰明白的看出数据的分布以及一些存在的问题，比如长尾分布等为后续的数据处理，数据填充提供了方向。

接下来就是推荐系统的主题部分了。推荐系统整体分为五大阶段：数据收集阶段，召回阶段，多路召回阶段，排序阶段，预测阶段。推荐系统算法这是我第一次接触，所以花费了许多的时间在算法思想的理解上。数据收集阶段的两种模式offline/online，更贴近于实际的应用场景，将模型训练阶段与模型部署阶段分成了两个部分

当然，本次结果我认为还有一些可以提升的地方。

1. 关于冷启动的问题解决方案

本次比赛我并没有关注冷启动方面的问题，并且并没有使用文章Embedding数据集去计算文章相似度，对于一些出现频率较少、甚至第一次出现的文章美能做到很好的预测。如果关注了冷启动这方面，可能会有更好的结果。

2. 关于模型融合的解决方案

在最后的模型融合方案中，并没有找到一个很好的解决方案去将各个模型形成的推荐文章进行整合，最终是由单模型产生结果，可能会有局限性。如果能将多种模型的结果进行整合，可能会有更好的结果。

3. 引入神经网络进行预测

现阶段有许多推荐系统是基于深度神经网络进行预测分析的，并且有许多论文进行支撑。或许将神经网络加入系统可以达到更好的预测效果。