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	沈晨玓	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。

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：沈晨珀　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

一、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

2．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。

数据表

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

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

3．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): $\mathrm{HR} = \frac{\#\mathrm{hit}}{\#\mathrm{user}}$

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

$$\mathrm{score(user)} = \sum_{k=1}^{5} \frac{\mathrm{s(user,k)}}{\mathrm{k}}$$

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）

4．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

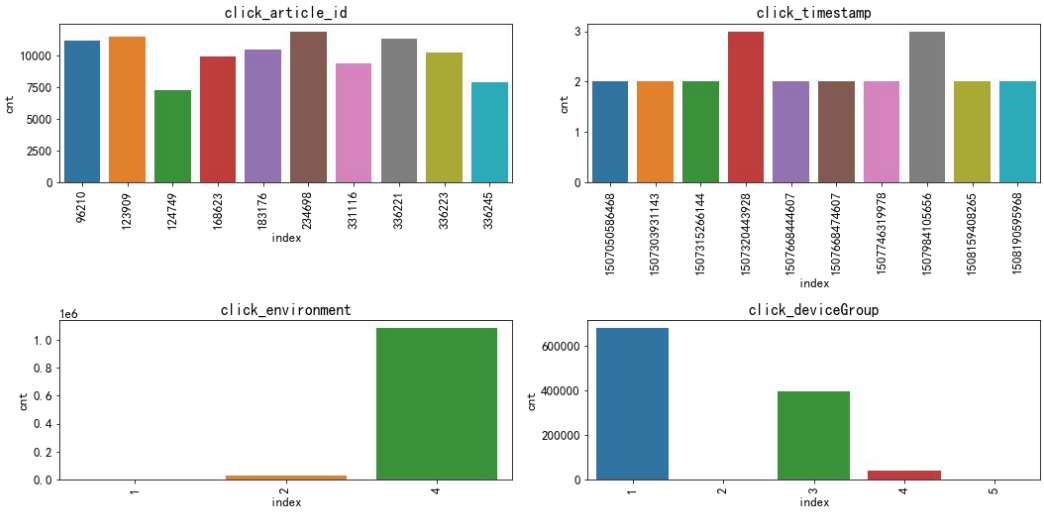
5．赛题分类

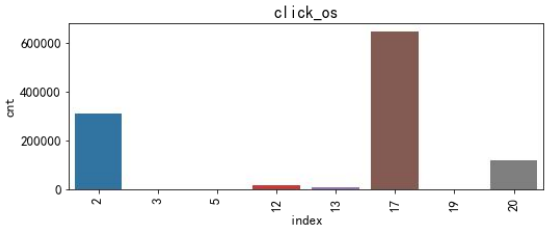
最终需要给用户从高到低推荐 5 篇文章进行预测，因此为排序问题，可以通过一定方式转换为二分类问题（后续会提及方法）。

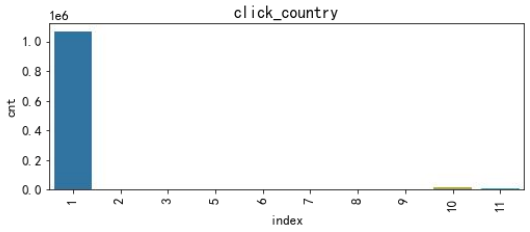
二、数据统计与可视化分析

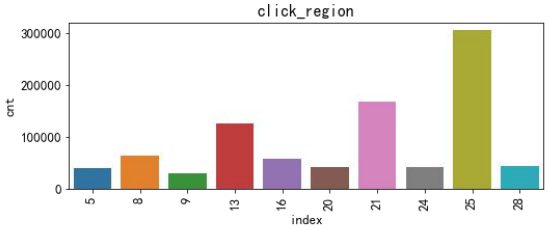
1．数据集可视化展示

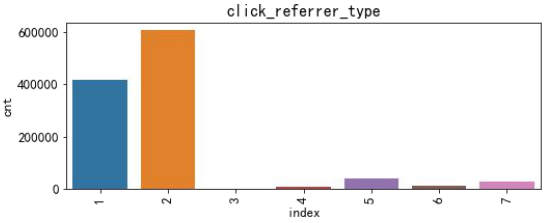
（1）Click_log（用户点击记录）

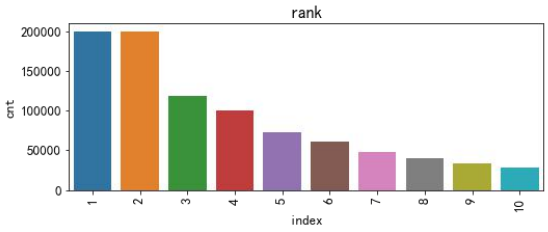


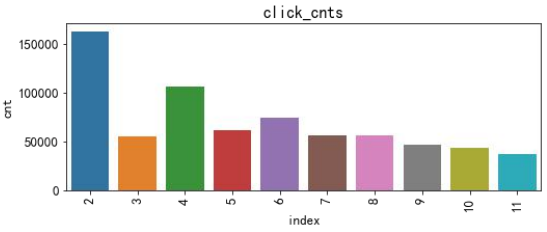












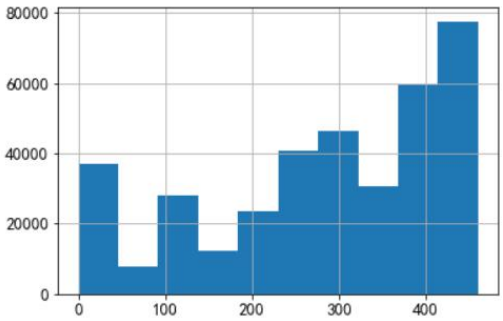
（2）Articles.csv（文章信息）

统计单词数量

```
176     3485
182     3480
179     3463
178     3458
174     3456
       ...
556        1
625        1
2798       1
767        1
841        1
Name: words_count, Length: 866, dtype: int64
```

统计文章主题

共 461 个主题，分布如下



（3）Article_emb.csv（文章向量）

	article_id	emb_0	emb_1	emb_2	emb_3	emb_4	emb_5	emb_6	emb_7	emb_8	...	emb_240
0	0	-0.161183	-0.957233	-0.137944	0.050855	0.830055	0.901365	-0.335148	-0.559561	-0.500603	...	0.321248
1	1	-0.523216	-0.974058	0.738608	0.155234	0.626294	0.485297	-0.715657	-0.897996	-0.359747	...	-0.487843
2	2	-0.619619	-0.972960	-0.207360	-0.128861	0.044748	-0.387535	-0.730477	-0.066126	-0.754899	...	0.454756
3	3	-0.740843	-0.975749	0.391698	0.641738	-0.268645	0.191745	-0.825593	-0.710591	-0.040099	...	0.271535
4	4	-0.279052	-0.972315	0.685374	0.113056	0.238315	0.271913	-0.568816	0.341194	-0.600554	...	0.238286

2．数据分析

（1）用户特征

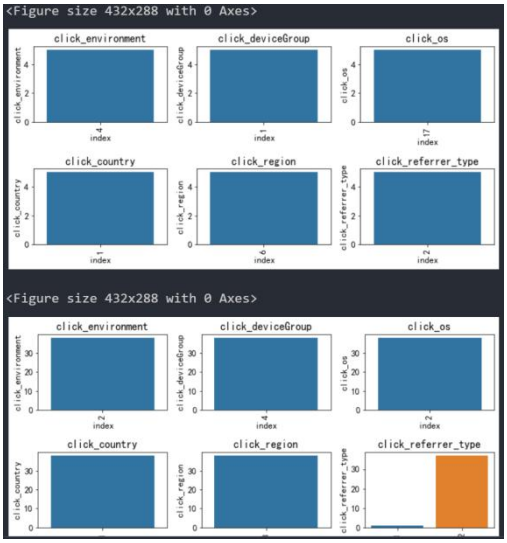
a．用户重复点击次数

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。

```
1      1605541
2        11621
3          422
4           77
5           26
6           12
10           4
7            3
13           1
Name: count, dtype: int64
```

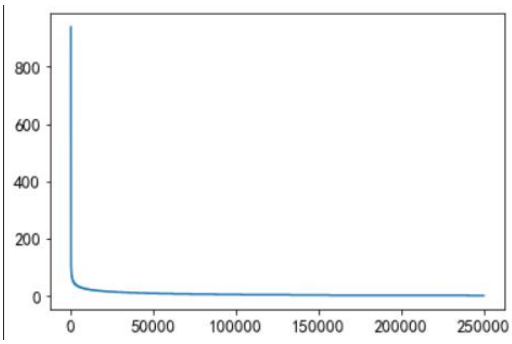
b．用户点击环境变化

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。



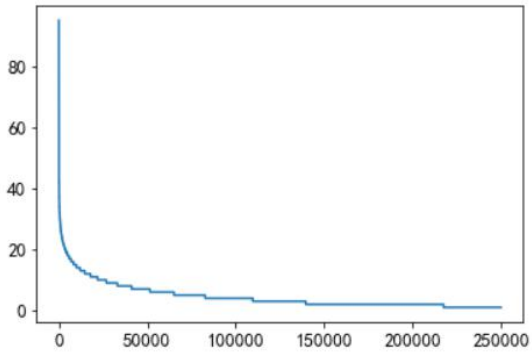
c．用户点击新闻总次数

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。



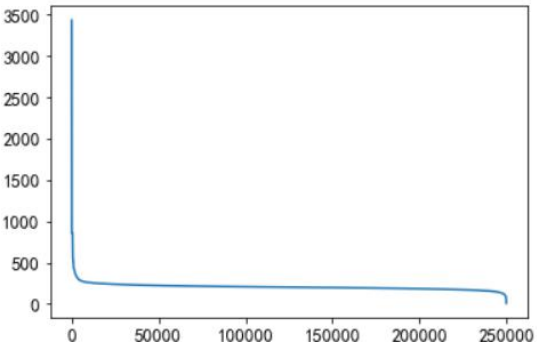
d．用户点击新闻类型统计

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。



e．用户查看新闻字数长度统计

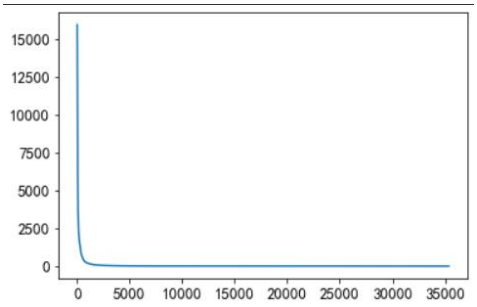
有一小部分人看的文章平均词数非常高，也有一小部分人看的平均文章次数非常低。大多数人偏好于阅读字数在 200-400 字之间的新闻。大多数人都是看 250 字以下的文章。

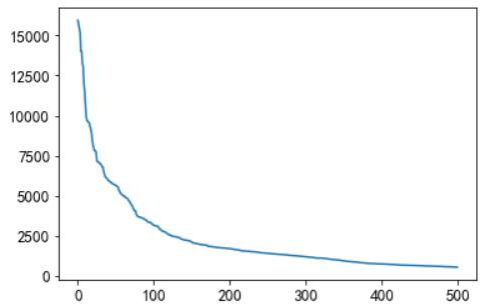


(2) 文章特征

a．新闻被点击总次数

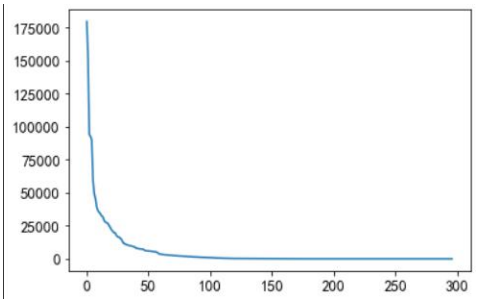
可以看到大部分新闻被点击总次数在 1000 次以下，可以以此为热门/不热门新闻的分界值。



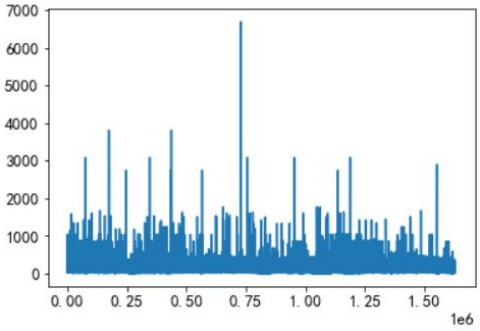


b．新闻具体特征

i）不同类型的新闻出现的次数

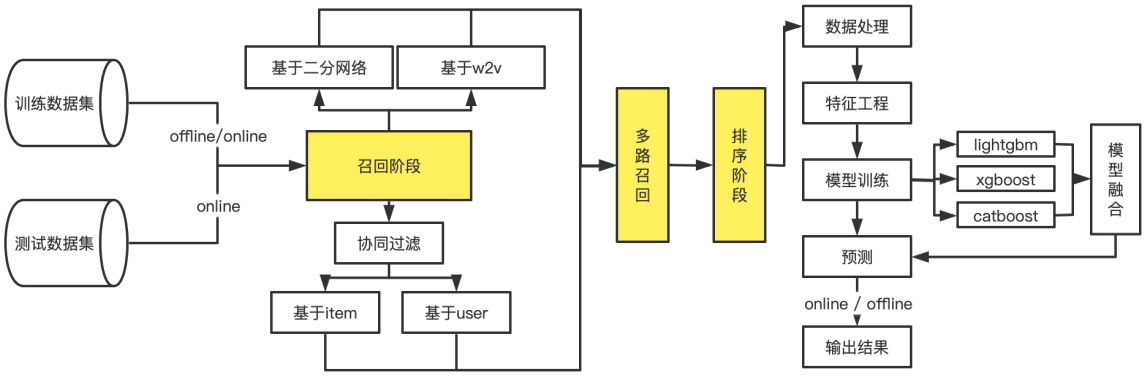


ii）文章字数的统计



三、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。

四、数据收集阶段

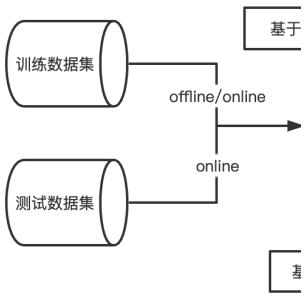
推荐系统的最终目的是完成对于线上用户的实时推荐任务，所以分为了离线/在线两种模式（offline/online）。

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。

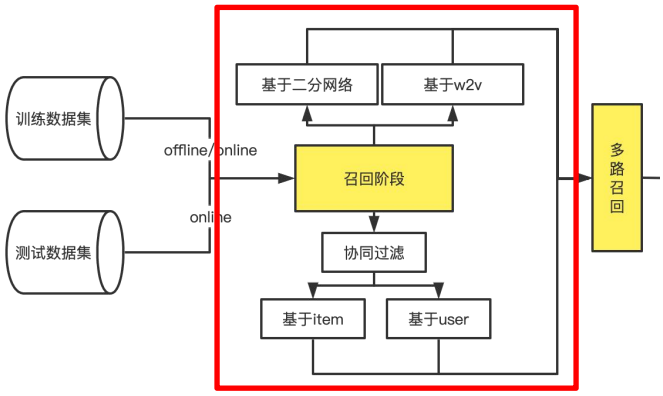
在下图中可以清晰的看到不同模式下的数据来源：



五、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

A. 基于物品的协同过滤（itemCF）

a) 整体思路：给用户推荐与曾经读过文章相似的文章

b) 步骤：

i. 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：

$$W_{ab} = \frac{|N(a) \cap N(b)|}{|N(a)|}$$

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：

$$W_{ab} = \frac{|N(a) \cap N(b)|}{\sqrt{|N(a)||N(b)|}}$$

***核心代码：2_recall_itemCF.py 中 cal_sum 函数**

ii. 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：

$$P_{ub} = \sum_{a \in N(u) \cap S(b,k)} W_{ab} R_{ua}$$

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**

iii. 根据P_{ub}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。

c) 创新点：

i. 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

ii. 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。

$$W_{ab} = \frac{|N(a) \cap N(b)|}{\sqrt{\log(|N(a)|) * \log(|N(b)|)}}$$

***核心代码**

```
sim_dict[item][relate_item] += loc_weight / \
    math.log(1 + len(items))
```

此时效果得到提升：HR_5 = 0.30320，MRR_5 = 0.17134

iii. 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**

```
# 可调参数
loc_alpha = 1.0 if loc2 > loc1 else 0.7
loc_weight = loc_alpha * (0.9**(np.abs(loc2 - loc1) - 1))

```

```
    sim_dict[item][relate_item] += loc_weight / \
        math.log(1 + len(items))
```

```
for loc, item in enumerate(interacted_items):
    for relate_item, wij in sorted(item_sim[item].items(), key=lambda d: d[1], reverse=True)[0:200]:
        if relate_item not in interacted_items:
            rank.setdefault(relate_item, 0)
rank[relate_item] += wij * (0.7**loc)
```

iv. 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**

```
interacted_items = interacted_items[::-1][:2]
```

d) 最终效果：

HR_5 = 0.32958，MRR_5 = 0.19303

B. 基于用户的协同过滤（userCF）

a) 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

b) 步骤：

i. 计算用户之间的相似度

ii. 根据用户的相似度和用户的历史行为给用户推荐

iii. 根据P_{ub}推荐 Top100 的文章给用户

c) 遇到困难：

i. 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。

ii. 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。

C. 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

a) 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。

b) 步骤：

i. 计算物品之间的相似度

ii. 根据物品的相似度和用户的历史行为给用户推荐

iii. 根据P_{ub}推荐 Top100 的文章给用户创新点：

c) 创新点：

i. 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

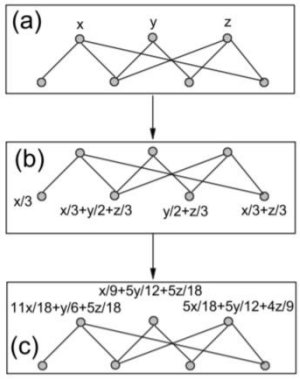


FIG. 2. Illustration of the resource-allocation process in bipartite network. The upper three are X nodes and the lower four are Y nodes. The whole process consists of two steps: First, the resource flows from X to Y ($a \to b$), and then returns to X ($b \to c$). Different from the prior network-based resource-allocation dynamics [53], the resource here can only flow from one node set to another without consideration of asymptotical stable flow among one node set.

ii. 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 0.19370，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。

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

***核心代码**

```
interacted_items = interacted_items[::-1][:1]
```

iii. 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**

```
for user in users:
    tmp_len = len(user_item_dict[user])
    for relate_item in user_item_dict[user]:
```

```
            sim_dict[item].setdefault(relate_item, 0)
            sim_dict[item][relate_item] += 1 / \
                ((len(users)+1) * (tmp_len+1))
```

d) 最终效果：

HR_5 = 0.35613，MRR_5 = 0.20928

D. 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

a) 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

b) 步骤：

i. 将文章列表通过 Word2Vec 转化为 256 维的向量。

ii. 根据文章向量计算文章相似度。

iii. 根据相似度计算推荐 Top100 的文章给用户

c) 论文核心思想：

任何物品都可以转换为向量，即 item2vec 的思想。

Since we ignore the spatial information, we treat each pair of items that share the same set as a positive example. This implies a window size that is determined from the set size. Specifically, for a given set of items, the objective from Eq. (1) is modified as follows:

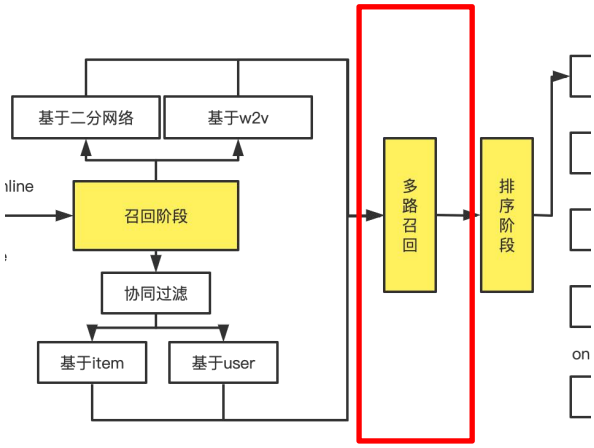
$$\frac{1}{K}\sum_{i=1}^{K}\sum_{j\neq i}^{K}\log p(w_j|w_i).$$

Another option is to keep the objective in Eq. (1) as is, and shuffle each set of items during runtime. In our experiments we observed that both options perform the same.

六、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**

```
weights = {'itemcf': 1, 'binetwork': 1, 'w2v': 0.1}
recall_result['sim_score'] = recall_result['sim_score'] * weight
```

```
# 合并召回结果
recall_final = pd.concat(recall_list, sort=False)
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

2. 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**

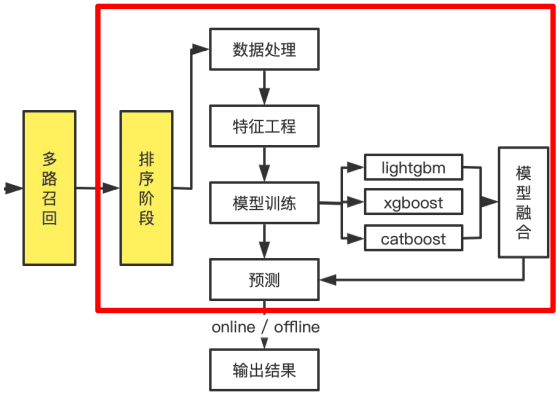
```
# 删除无正样本的训练集用户
gg = recall_final.groupby(['user_id'])
useful_recall = []

for user_id, g in tqdm(gg):
    if g['label'].isnull().sum() > 0:
        useful_recall.append(g)
    else:
        label_sum = g['label'].sum()
        if label_sum > 1:
            print('error', user_id)
        elif label_sum == 1:
            useful_recall.append(g)
```

最终每个用户平均召回 151 篇文章。

七、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。

A. 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：

i. 新闻特征

- 新闻字数
- 新闻创建时间
- 新闻被阅读数量

ii. 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

iii. 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**

B. 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。

a) LGBMClassifier

经测试，已知最佳模型参数：

```
model = lgb.LGBMClassifier(num_leaves=80,
                           max_depth=9,
                           learning_rate=0.01,
                           n_estimators=100000,
                           subsample=0.8,
                           feature_fraction=0.8,
                           reg_alpha=0.5,
                           reg_lambda=0.5,
                           random_state=seed,
                           importance_type='gain',
                           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

```
2021-12-15 18:32:50,368 - 7_rank_lgb.py[line:121] - DEBUG: importance:
0          user_last_click_article_binetwork_sim  4.547441e+06
1               user_last_click_article_w2v_sim  1.553009e+06
2                                     sim_score  1.494159e+06
3                user_last_click_timestamp_diff  7.869570e+05
4             user_last_click_article_itemcf_sim  6.589831e+05
5                       user_id_category_id_cnt  5.459145e+05
6           user_clicked_article_itemcf_sim_sum  4.659359e+05
7             user_last_click_created_at_ts_diff  2.140948e+05
```

b) XGBClassifier

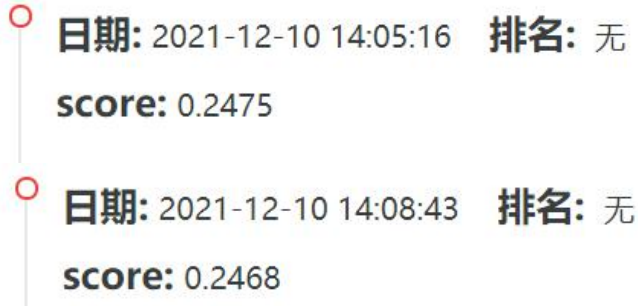
经测试，已知最佳模型参数：

```
model = xgb.XGBClassifier(max_depth=10,
                          learning_rate=0.08,
                          num_leaves=64,
                          n_estimators=1000000000,
                          subsample=0.8,
```

```
feature_fraction=0.75,
reg_alpha=0.7,
reg_lambda=1.2,
random_state=seed,
eval_metric='AUC',
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

```
2021-12-11 12:27:34,598 - 7_rank_xgb.py[line:114] - DEBUG: importance:
0                                    sim_score    0.200532
1          user_last_click_article_binetwork_sim    0.116839
2               user_last_click_article_w2v_sim    0.091702
3                user_last_click_timestamp_diff    0.043304
4           user_last_click_article_itemcf_sim    0.035041
5                                article_id_cnt    0.030459
6           user_last_click_created_at_ts_diff    0.025257
```

c) CatBoostClassifier

经测试，已知最佳模型参数：

```
model = cbt.CatBoostClassifier(max_depth=16,
                               learning_rate=0.08,
                               n_estimators=10000,
                               subsample=0.75,
                               random_state=seed,
                               eval_metric='AUC',
                               bootstrap_type='Poisson',
                               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

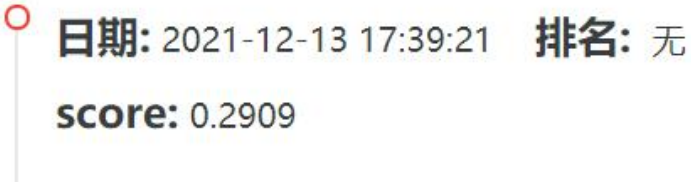
测试完了三种单模型预测，我尝试进行了三种模型的融合。

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：

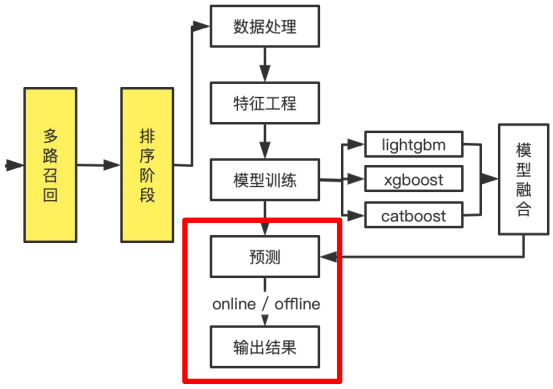
方案二：将所有模型中得分最高的三份结果进行整合合并。将共同推荐的文章放置在最前，其余位置按照三者之中最高分模型剩余文章进行填充。（由于 LGBM 模型得分远远高于其他两种模型，因此三者均为 LGBMClassifier）

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。

八、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%

赛季2	奖金	参赛队伍
2022-02-28	¥0	7712

长期赛: 17 /0.2914

长期赛 正式赛

排名	参与者	组织	score	最优成绩提交日
1	推荐算法3	厦门大学	0.3073	2021-04-08
2	sunjkk6	北京邮电大学	0.3035	2021-05-12
3	西西动动	清华大学	0.3029	2021-05-31
4	yisuwang666	北京邮电大学	0.3025	2021-05-11
5	_小彭同学_	华南理工大学	0.3004	2021-02-16
6	如何成为一名富婆	Others-Others	0.2988	2021-05-31
7	原麦山丘hy	清华大学	0.2988	2021-05-30
8	零度不含糖1	aa	0.2959	2021-05-26
9	零度不含糖	金融机构	0.2947	2021-05-29
10	一只鲸鲸鲸鲸	上海交通大学	0.2936	2021-06-02
11	zhx99	z	0.2935	2021-07-28
12	xixixixixxxxx	上海交通大学	0.2919	2021-06-01
13	xbingbingo	复旦大学	0.2918	2021-06-14
14	csphillipmassa	江西财经大学	0.2915	2021-04-24
15	漫天猪头飞	东北大学	0.2914	2021-03-22
16	sekkkkey	*	0.2914	2021-11-21
17	SZU沈晨玓	深圳大学	0.2914	2021-12-13
18	pkqpkq	西安电子科技大学	0.2914	2021-11-16

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。

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。