**基于大数据的用户行为分析及个性化推荐案例**

**——基于物品的电子商务商品推荐**[[1]](#footnote-1)

**摘 要**

基于内容的协同过滤推荐系统通过分析用户的历史行为，主动向用户推荐能够满足他们兴趣和需求的信息，帮助用户发现其感兴趣的物品，节约时间，提升用户体验、对电子商务网站的忠诚度。本案例依据电子购物网站大量数据，研究用户的兴趣偏好，分析用户的需求和行为，发现用户的潜在兴趣点，从而引导用户发现自己的信息需求，帮助用户发现他们感兴趣的信息，为用户提供个性化的服务。

**关键词：**推荐系统 协同过滤 数据分析 数据挖掘 电子商务

**Abstract**

By analyzing users' historical behaviors, the content-based collaborative filtering recommendation system actively recommends information that can surffice their interests and needs, helps users find items they are interested in and save their time, improves user experience and loyalty to e-commerce websites.Based on massive data of shopping website, this case studies users' interest preferences, analyzes users' demands and behaviors, and finds users' potential interest points, guides users to find their information needs, and helps users find the information they are interested in, provides users with personalized services.

**Keywords:** recommendation system; collaborative filtering; data analysis; data mining; e-commerce

**0 引 言**

随着互联网和信息技术的快速发展，电子商务、网上服务与交易等网络业务越来越普及，大量的信息聚集起来形成海量信息。用户想要从海量信息中快速准确地寻找到自己感兴趣的信息已经变得越来越困难，在电子商务领域这点显得更加突出。因此，信息过载的问题已成为互联网技术中的一个重要难题。

**1背景介绍**

互联网公司在日常运营中生成、累积的用户网络行为数据。一天之中，互联网产生的全部内容可以刻满1.68亿张DVD；发出的邮件有2940亿封之多（相当于美国两年的纸质信件数量）；发出的社区帖子达200万个（相当于《时代》杂志770年的文字量）；卖出的手机为37.8万台，高于全球每天出生的婴儿数量37.1万。从2012年开始，全球数据量已经从[TB](https://baike.baidu.com/item/TB)（1024GB=1TB）级别跃升到[PB](https://baike.baidu.com/item/PB/399855%22%20%5Ct%20%22_blank)（1024TB=1PB）、[EB](https://baike.baidu.com/item/EB/400198%22%20%5Ct%20%22_blank)（1024PB=1EB）乃至[ZB](https://baike.baidu.com/item/ZB/400300%22%20%5Ct%20%22_blank)(1024EB=1ZB)级别。

在互联网时代发展的前期，为了应对网络数据的增长催生了搜索引擎和网址分类网站，典型代表是谷歌和雅虎。但随着为了正因为数据量过大现象的越发明显，用户在搜寻查找自己想要的精确信息时变得越来越困难，在电子商务领域这点显得更加突出。因此，信息过载的问题已成为互联网技术中的一个重要难题，

也是在此背景下催生了推荐系统和推荐相关算法的发展，希望能够通过网站对用户的行为数据自主学习，从而为用户发掘出他们喜欢的内容，发现用户的潜在兴趣爱好内容。

**2 相关知识及技术介绍**

推荐系统的主要算法可分为两大类:

一、协同过滤算法；二、基于内容数据的过滤算法。基于内容数据的算法适用于冷启动，即用户和项目都比较少的时候，而基于用户行为数据的协同过滤算法在用户和项目较多，数据比较丰富的情况下有较高的准确率。

除此之外，还包括基于社会网络数据的推荐，基于语境（上下文）感知数据的推荐，基于心理学数据的推荐等等。

**2.1. 协同过滤算法(collaborative filtering)**

协同过滤推荐算法的基本假设是:为用户推荐感兴趣的内容可通过找到与该用户偏好相似的其他用户，将他们感兴趣的内容推荐给该用户。算法可以概括为两步:①采用相似度计算方法计算用户/项目之间的相似度，构造相似度矩阵;②采用相应的算法估计评分，并据此为用户进行推荐。该类方法可为目标用户估计对某一特定项目的评分，也可产生一个推荐列表。进一步可分为基于基于用户的协同过滤算法（user-based CF）和基于内容（项目、物品）的协同过滤（item-based CF）。

**2.1.1 基于用户的协同过滤算法（user-based CF）**

基于用户(user-based)的推荐算法为目标用户$U\_{i}$(i=1，2，…，n)估计给定项目$I\_{j}$ (j=1，2，…，m)的评分$ p\_{i,j}$。该方法首先计算用户$U\_{i}$和其他用户间的相似度，选取其他用户中为$I\_{j}$评过分的用户构成集合$U\_{i}\*$，根据所有$U\_{k}\in U\_{i}\*$对项目$I\_{j}$的评分来估计$U\_{i}$对$I\_{j}$的评分。一个用户喜欢和他具有相似喜好的用户喜欢的项目，两个用户喜欢的项目交集越大，这两个用户越相似。基于用户的推荐方法面对用户数量不断变化的系统，需要经常重新计算用户间的相似度矩阵，因而时间复杂性高，可扩展性较差。

**2.1.2 基于内容的协同过滤（item-based CF）**

基于内容的推荐算法为目标项目$I\_{i}$ (i=1，2，…，n)估计给定用户$U\_{j}$ (j=1，2，…，m)的评分$ p\_{i,j}$。该方法首先计算项目$I\_{i}$和其他项目间的相似度，选取其他项目中为$U\_{j}$评过分的项目构成集合$U\_{i}\*$，根据所有$U\_{k}\in U\_{i}\*$对用户$U\_{j}$的评分来估计$I\_{i}$对$U\_{j}$的评分。它基于这样的假设：一个用户会喜欢与他之前喜欢的内容相似的项目。因此，基于内容的协同过滤推荐关键在于计算物品之间的相似度。

协同过滤推荐算法需要维护一个用户相似度矩阵或内容相似度矩阵，因此对于内容的数目更新速度远远小于用户数目的增长速度的情况，宜采用基于内容的推荐算法。基于内容的算法一般在性能上要优于基于用户的算法。

**2.1.3相似度计算方法**

两个用户兴趣相似度的计算可以有多种方法，常见有Jaccard相似系数, Pearson相关系数和夹角余弦相似度计算。

1. **Jaccard相似度**

$$J\left(A\_{1},A\_{M}\right)=\frac{\left|A\_{1}\bigcap\_{}^{}A\_{M}\right|}{\left|A\_{1}\bigcup\_{}^{}A\_{M}\right|}$$

分母$A\_{1}\bigcup\_{}^{}A\_{M}$表示喜欢的物品1（用户）与物品（用户）M的用户总数，分子$A\_{1}\bigcap\_{}^{}A\_{M}$表示同时喜欢物品（用户）1与物品（用户）M的用户数。

1. **Pearson相关系数**

$$sim\left(x,y\right)=\frac{\sum\_{i\in I\_{x,y} }^{}( r\_{x,i}-\overbar{r\_{x}}) ( r\_{y,i}-\overbar{r\_{y}})}{\sqrt{\sum\_{i\in I\_{x,y} }^{}\left( r\_{x,i}-\overbar{r\_{x}}\right)^{2} \sum\_{i\in I\_{x,y} }^{}\left( r\_{y,i}-\overbar{r\_{y}}\right)^{2} }}$$

其中 ，$ I\_{x,y}$为用户 x 和 y 共同评过分的项目集合，$ r\_{x,i}$表示用户 x 对项目i 的评分。$\overbar{r\_{x}}$表示用户 x 的平均评分。相关系数取值范围是[-1,1]，当余弦值接近±1，表明两个向量具有较强的相似性。当余弦值为0时表示不相关。

1. **夹角余弦相似度**

$$sim\left(x,y\right)=\frac{\sum\_{i\in I\_{x,y} }^{}r\_{x,i} r\_{y,i}}{\sqrt{\sum\_{i\in I\_{x,y} }^{}r\_{x,i}^{2} \sum\_{i\in I\_{x,y} }^{}r\_{y,i}^{2} }}$$

其中,$ I\_{x,y}$为用户 x 和 y 共同评过分的项目集合,$ r\_{x,i}$表示用户 x 对项目i 的评分。$sim\left(x,y\right)$取值范围是[-1,1]，相关系数的绝对值越大，则表明两者相似性越高。

**2.3 基于模型的协同过滤（model-based CF）**

基于模型的协同过滤推荐是采用机器学习或数据挖掘等算法，用训练数据来学习识别复杂模式，从而得到学习模型，然后基于学习模型在数据集上进行智能预测。主要有以下模型：

* 隐语义模型/矩阵分解模型
* 贝叶斯信念网协同过滤模型
* 聚类协同过滤模型
* 概率模型

**2.4 基于内容数据的推荐**

对一个给定的用户，推荐与他之前喜欢的内容在内容上有相似性的其他项目。这种推荐仅需要得到两类信息：内容特征的描述和用户过去的喜好信息。利用领域专家给内容打标签的方法，也即传统的分类系统，另一种是用户给内容打标签，也即大众分类系统。

**2.5 基于文本语境感知数据的推荐**

语境信息类型包括时间、信息、外界物理环境（如天气、温度等）、设备类型、周围人员、活动状态、目的/意图等。还有些系统考虑了情绪、计算平台、网络条件、社会网络等更为广泛的语境。

**2.6 推荐系统的评价指标**

推荐系统的评价指标有预测准确度、覆盖率、召回率、F1分数（score）、受试者工作特征曲线（ROC）等。其中准确度是最重要也是最常用的推荐系统评测指标，用来度量推荐系统预测的能力。对于top-M推荐，用准确率（precision）召回率（recall）和F1分数来评测推荐系统。

根据不同的推荐模型应当选择不同的模型评测指标，常用的评测指标模型见下表1：

 **表1 评测指标表**

|  |  |  |  |
| --- | --- | --- | --- |
| 数据表现方式 | 模型1 | 模型2 | 模型3 |
| 预测准确度 | $RMSE= \sqrt{\frac{1}{N}\sum\_{}^{}\left( r\_{u,i}-\overbar{r\_{u}}\right)^{2}}$  | $$MAE= \frac{1}{N}\sum\_{}^{}\left| r\_{u,i}-\overbar{r\_{u}}\right|$$ | $precision= \frac{TP}{TP+FP}$ |
| 分类准确度 | $precision= \frac{TP}{TP+FP}$  | $recall= \frac{TP}{TP+FN}$  | $F1= \frac{TP}{TP+FN}$  |

由于用户行为存在不同，例如加入购物车、是否购买、是否评论、评论的内容、点赞等等不同，衡量表示所有行为喜好程度的评分机制也会有所不同。在某些网站的物品需要对其进行评分，因而需要使用的预测准确度评分方式：均方根误差（RMSE），平均绝对误差（MAE）。本例中的数据为用户浏览过的商品，行为可划分为二元选择（浏览过标记为1，未浏览过标记为0），因此可以选用模型3进行衡量。

**3.实施方案**

**本案例可以分为以下几个步骤：**

**（1）数据预处理**

1. 导入pandas库打开数据集，浏览原始数据
2. 删除无用项 （没有评分的行、空值NA的列）
3. 更改列名、重新编号编号以便于后续数据处理

**（2）数据分析与处理**

其主要步骤如下：

1. 把处理后数据变成偏好二元矩阵
2. 统计总用户数、总物品数、总用户行为（有评分）记录数
3. 用户评价物品所占物品总数比例统计
4. 各物品被点击次数
5. 商品热门指数统计

**（3）协同过滤算法实现**

1. **基于物品的协同过滤算法步骤**
2. 构建偏好二元矩阵P；
3. 用相关性度量方法计算不同物品间的相似度，构建物品相似度矩阵S；
4. 对于用户u，若其偏好矩阵$ P\_{u}$元素之和为零，即没有喜欢的物品（冷启动问题），采用其他推荐方法：随机推荐或热门（Popular）算法;
5. 若其偏好矩阵$ P\_{u}$元素之和不为零，针对其偏好矩阵中感兴趣的物品i，分别从相应相似矩阵$ S\_{i}$中选择相似度最高的K个物品，并按如下公式计算用户u对一个物品j的兴趣：

 $P\_{u,j}=\sum\_{i\in N(u)\bigcap\_{}^{}S(j,M)}^{}w\_{j,i}r\_{u,i}$

其中，$N(u)$为用户喜欢的物品的集合，$S(j,M) $是和物品j最相似的M个物品的集合，$r\_{u,i}$表示用户u对物品i的兴趣（对于偏好矩阵为二元矩阵即$P\_{u,j}$只有0或1的值时，可令$r\_{u,i}=1$）；

1. 选择评估模型，对于预测结果进行评估。

**b.代码实现算法主要步骤**

1. 对处理后的数据构建二元偏好矩阵P；
2. 用Jaccard相似系数计算不同物品间的相似度，构建物品相似度矩阵S；
3. 对于用户u，若其偏好矩阵$ P\_{u}$元素之和为零，即没有喜欢的物品（冷启动问题），采用其他推荐方法：随机推荐或Popular算法;
4. 定义函数similarity，用于计算一个用户每个喜欢的物品对应推荐k个物品，返回列表（二元组结构）；定义函数predict，用于计算一个推荐物品的预测值，返回列表（二元组结构）；
5. 用函数similarity计算一个用户每个喜欢的物品对应推荐k个物品，分别用函数predict计算出每个推荐物品的兴趣值；
6. 选出预测兴趣度最高M个物品（为了便于处理，设k=M）；
7. 模型评估：用准确度、召回率、F1分数（F1 score）作为模型评估指标；
8. 交叉验证：使用sklearn.model\_selection.ShuffleSplit将数据集随机打乱，并将原始数据按7/3比例划分为训练集/测试集，并迭代25次，模型评估指标去测试结果的平均值；
9. 模型效果对比：同时运用随机算法对比协同过滤算法。
10. 选择为（每个用户推荐物品）不同的M值，并分别记录不同的M值下模型评估指标值

**c.其他算法实现：随机推荐、热门推荐**

**（4）评估预测结果**

绘制不同的M值、不同推荐算法下的准确度、召回率、F1分数图表，对比选出最优模型用于推荐系统。

**4用户行为分析与推荐**

**4.1实验环境**

本案例以Python3.6为工具进行数据的获取、预处理、建模与分析工作。实验环境如下:

操作系统：Windows 7

处理器：2.50GHz 双核intel i5

内存：4GB

Python版本: 3.6

Python编辑器：Jupyter Notebook

Python包：pandas, numpy , sklearn, matplotlib, urllib，random,nltk等。

**4.2数据获取**

本案例利用urllib等包从电子商务网站Retailrocket爬取用户对数个月时间内某分类下150个物品的用户行为，利用爬虫获取数据的流程见附件图1所示，并将获取的数据保存为csv格式，并利用pandas进行数据解析。数据的属性包括：用户名称、物品名称、用户行为（浏览该物品）。

打开的部分原始数据如下：

* 1. **数据清洗**

提取喜欢物品用户（“1”）；去掉无用标记“0”；列重命名

train**.**drop**([**"Unnamed: 0"**,**"2"**,**'4'**],**axis**=**1**,**inplace**=True)** #删除无用列

train\_1**=**train

train\_1**=**np**.**array**(**train\_1**)**

train\_1**=**train\_1**[**train**.**loc**[**train**[**'5'**]==**1**].**index**.**tolist**(),::]**

frame**={**'userid'**:**train\_1**[::,**0**],**'itemid'**:**train\_1**[::,**1**],** 'ratings'**:**train\_1**[::,**2**]}**#列重命名

train**=**pd**.**DataFrame**(**frame**)**

用户物品从0开始编号：

train**[**'userid'**]=**train**[**'userid'**]-**1

train**[**'itemid'**]=**train**[**'itemid'**]-**1

处理结果（部分显示）：

构建偏好矩阵：

a**=**train**[**"userid"**].**max**()+**1#用户数

b**=**train**[**"itemid"**].**max**()+**1#物品数

pref**=**np**.**zeros**((**a**,**b**))**#初始化

train\_1**=**np**.**array**(**train**)**

**for** i **in** range**(**len**(**train**)):**

 **if** train\_1**[**i**,**2**]==**1**:**

 pref**[**train\_1**[**i**,**0**],**train\_1**[**i**,**1**]]=**1

* 1. **数据可视化及分析**

**print(**'用户数：%d'**%(**a**))**

**print(**'物品数：%d'**%(**b**))**

**print(**'有效记录数：%d'**%(**len**(**train**)))**

用户数：63978

物品数：150

有效记录数：1169658

由此可看出用户数远大于物品数，由之前的分析可知，使用基于内容的协同过滤推荐算法更合适，效果更好。

用户行为记录所占物品总数比例，由此可进一步挖掘出新老顾客的比例，

pd.DataFrame(pref.sum(axis=1)/b)

pref.sum(axis=0)#各物品被点击次数

商品热门指数，越接近1越热门；

pref.sum(axis=0)/a

按热门程度物品分布比例

data\_view = pd.DataFrame(pref.sum(axis=0)/a)

data\_view.columns=['item']

bins = [-1,0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]

data\_view['item\_bins'] = pd.cut(data\_view['item'],bins)

(data\_view.item\_bins.value\_counts()/b).plot.bar()

plt.xlabel('商品热门指数')

plt.ylabel("占物品总数比例")

**图1商品热门指数分布图**

从商品的热门指数分布可以看出，最热门的前20%商品获得了非常高的点击率；当遇到冷启动问题时，据此可以将分布于前20%的商品用于热门推荐。

* 1. **推荐算法构造实现**

根据上述分析可知，对于用户的行为，可能存在的情况：用户没有购物行为，用户具有购物行为。对于没有购物行为的用户，如果是用协同过滤推荐会产生冷启动问题，产生的性能不佳，对此可以使用随机推荐或热门推荐。据此，构建三种推荐算法，基于内容的协同过滤推荐算法，随机推荐和热门推荐算法；在有用户行为的用户中，对基于内容的协同过滤推荐算法，随机推荐和热门推荐算法进行性能比较。推荐算法实现如下：

1. 随机推荐

随机产生M个不同的物品编号。

 **import** random

**def** random\_select**(**a**,**M**):**#a,物品数;M，推荐的物品数

 temp\_list**=[]**

 **while** len**(**temp\_list**)<**M**:**

 temp**=**random**.**randint**(**0**,**a**-**1**)**

 **if** temp **not** **in** temp\_list**:**

 temp\_list**.**append**(**temp**)**

 **return** temp\_list

1. 基于内容的协同过滤推荐算法，
2. 用Jaccard相似系数计算不同物品间的相似度，构建物品相似度矩阵S；

**def** jaccard**(**a**,**b**):** #物品a与b的杰卡德相似度，仅对0-1矩阵有效

 **return** **(**a**\***b**).**sum**()/(**a**+**b**-**a**\***b**).**sum**()**

**def** sim\_matrix**(**x**):**#建立物品相似矩阵

 sim**=**np**.**ones**((**np**.**size**(**x**,**axis**=**1**),**np**.**size**(**x**,**axis**=**1**)))**

 **for** i **in** range**(**np**.**size**(**x**,**axis**=**1**)):**

 **for** j **in** range**(**np**.**size**(**x**,**axis**=**1**)):**

1. 对于用户u，若其偏好矩阵$ P\_{u}$元素之和为零，即没有喜欢的物品（冷启动问题），采用其他推荐方法：随机推荐或Popular算法;
2. 定义函数similarity，用于计算一个用户每个喜欢的物品对应推荐k个物品，返回一个列表（二元组结构）；

 **def** similarity**(**i**,**pref**,**sim**,**k**):** #i喜欢的每个物品最相似的k个物品集合

 L1**=[]**

 L3**=[]**

 a**=**np**.**size**(**pref**,**axis**=**0**)**#用户数

 b**=**np**.**size**(**pref**,**axis**=**1**)**#物品数

 **for** j **in** range**(**b**):**#i喜欢物品j最相似k个物品集合

 L2**=[]**#存放相似的物品及相似度

 l**=**0

 **if** pref**[**i**,**j**]==**1**:**

 L1**=**sim**[**j**,::]**

 L1**=**L1**.**tolist**()**

 **while(**l**<**k**):**

 a**=**L1**.**index**(**max**(**L1**))**

 **if** a**!=**j **and** pref**[**i**,**a**]==**0**:**

 L2**.**append**(**a**)**

 b**=**np**.**max**(**L1**)**

 L2**.**append**(**b**)**

 l**+=**1

 L1**[**a**]=-**9

 L3**.**append**(**L2**)**

 **return** L3

定义函数predict，用于计算一个推荐物品的预测值，返回列表（二元组结构）：

 **def** predict**(**a**,**b**,**L**):**#计算各推荐物品预测兴趣度;a,推荐物品；b,相似度 L:推荐物品名集合

 **for** i **in** range**(**len**(**L**)):**

 **if** a**==**L**[**i**][**0**]** **and** len**(**L**[**i**])==**1**:**

 L**[**i**].**append**(**b**)**

 **elif** a**==**L**[**i**][**0**]:**

 L**[**i**].**append**(**b**)**

 L**[**i**][**1**]=**L**[**i**][**1**]+**L**[**i**][**2**]**

 L**[**i**].**pop**(**2**)**

 **return** L

用set()类型定义集合temp\_predict，用于存放推荐物品的集合：

 temp\_predict**=**set**([])**

predict\_list**=[]**#用于该用户预测所有推荐物品兴趣度

 sim\_item**=**similarity**(**i**,**L**,**sim**,**k**)**

若为空即说明遇到冷启动问题，该用户没有感兴趣物品，应用其他方法推荐：

**if** len**(**sim\_item**)==**0**:**

**return** predict\_list

构建推荐物品的二元组temp\_predict，输出格式：[[a],...,[b]]：

 **for** j **in** range**(**len**(**sim\_item**)):**#所有可能预测的物品ID集合

 **for** k **in** range**(**0**,**len**(**sim\_item**[**j**]),**2**):**

 temp\_predict**.**add**(**sim\_item**[**j**][**k**])**

 temp\_predict**=**list**(**temp\_predict**)**

 temp**=[]**

 **for** j **in** temp\_predict**:**

 temp**.**append**([**j**])**

 temp\_predict**=**temp

根据$P\_{u,j}=\sum\_{i\in N(u)\bigcap\_{}^{}S(j,M)}^{}w\_{j,i}r\_{u,i}$计算每个物品预测评分，由于是二元矩阵，可令$r\_{u,i}=1$，故对于用户不同感兴趣物品推荐出的相同物品，只需要将它们的相似度相加即可。

 **for** j **in** range**(**len**(**sim\_item**)):**

 **for** k **in** range**(**0**,**len**(**sim\_item**[**j**]),**2**):**

temp\_predict**=**predict**(**sim\_item**[**j**][**k**],**sim\_item**[**j**][**k**+**1**],**temp\_predict**)**#输出格式：[[a,0.xx],...[b,0.xx]]

选出预测兴趣度最高M个物品

 **for** i **in** range**(**M**):**

 temp\_predict**=**np**.**array**(**temp\_predict**)**

 a**=**np**.**argmax**(**temp\_predict**,**axis**=**0**)**

 predict\_list**.**append**(**temp\_predict**[**a**[**1**]][**0**])**

 temp\_predict**=**np**.**delete**(**temp\_predict**,[**a**[**1**]],**axis**=**0**)**

最后返回 predict\_list ，预测物品输出格式：[[1,2,3],...,[x,y,z]]。

注意到整个过程中传入了两个参数k和M，一个是为一用户感兴趣每个物品推荐k个物品，一个是最终预测为用户推荐的M个物品，一般而言取值上M应当大于等于k，为便于分析取k=M。

1. 交叉验证：使用sklearn.model\_selection.ShuffleSplit将数据集随机打乱，并将原始数据按7/3比例划分为训练集/测试集，并迭代25次，模型评估指标去测试结果的平均值。主要代码如下：

**from** sklearn**.**model\_selection **import** ShuffleSplit

**import** random

rs**=**ShuffleSplit**(**n\_splits**=**25**,**test\_size**=**0.3**,**random\_state**=**0**)**#随机采样迭代25次，训练/测试数据集7/3划分

 rs**.**get\_n\_splits**(**X**)**

 **for** train\_index**,**test\_index **in** rs**.**split**(**X**):**#取多次测试结果平均值

 X\_train**,**X\_test**=**X**[**train\_index**],**X**[**test\_index**]**

 ran\_num**=**random**.**randint**(**0**,**b**-**1**)**#随机数

 **while** X\_test**[::,**ran\_num**].**sum**()==**0**:**

 ran\_num**=**random**.**randint**(**0**,**b**-**1**)**

 temp**=**pd**.**DataFrame**(**X\_test**[::,**ran\_num**])**

 index**=**temp**.**loc**[**temp**[**0**]==**1**].**index**.**tolist**()**

 X\_test**=**X\_test**[**index**,::]**#提取喜欢该物品用户

 X\_test**[::,**ran\_num**]=**0#用户矩阵中物品ran\_num标记为零，做测试

 a**=**np**.**size**(**X\_test**,**axis**=**0**)**#用户数

 sim**=**sim\_matrix**(**X\_train**)**

 pred\_rec**,**pred\_ran**=[],[]**

 **for** i **in** range**(**a**):**#推荐算法预测物品

 pred\_rec**.**append**(**Recommend**(**X\_test**,**sim**,**i**,**k**,**M**))** #协同过滤

 pred\_ran**.**append**(**random\_select**(**b**-**1**,**M**))** #随机推荐

1. 模型效果对比：同时运用随机算法对比协同过滤算法进行评估。
2. 热门推荐算法

顾名思义，即为某用户从过往点击率最高且该用户未产生过行为（浏览点击查看、评分等）的商品中选出若干个推荐给该用户。

 **def** popular\_select**(**L**,**i**,**M**):**#L:偏好矩阵；i,用户;M，推荐的物品数

 pre\_list**=[]**

 temp**=**pd**.**DataFrame**(**L**.**sum**(**axis**=**0**))**

 temp**=**temp**.**sort\_values**(**by**=[**0**],**ascending**=False).**index

 temp**=**np**.**array**(**temp**).**tolist**()**

 l**,**j**=**0**,**0

 **while(**l**<**M**):**

 **if** L**[**i**,**temp**[**j**]]==**0**:**

 l**+=**1

 pre\_list**.**append**(**temp**[**j**])**

 j**+=**1

 **return** pre\_list

* 1. **模型评估分析**

从之前的表1可知，由于用户行为存在不同，例如加入购物车、是否购买、是否评论、评论的内容、点赞等等不同，衡量表示所有行为喜好程度的评分机制也会有所不同。在某些网站的物品需要对其进行评分，因而需要使用均方根误差（RMSE）或平均绝对误差（MAE）的预测准确度评分方式。而本例中的数据为用户浏览过的商品，行为可划分为二元选择（浏览过标记为1，未浏览过标记为0），因此可以选用模型3，即用准确度、召回率、F1分数（F1 score）作为模型评估指标进行衡量。

 **def** P\_R\_F1Score**(**test**,** predict**,**M**):**#分别计算Precision，Recall，F1 score

 hit**,**n\_recall\_num**,**n\_precision **=**0**,**0**,**0

 a**=**len**(**predict**)**#用户数

 **if** a**==**0**:**

 **return** **[**0**,**0**,**0**]**

 **else:**

 **for** i **in** range**(**a**):**

 **if** test **in** predict**[**i**]:**

 hit**+=**1

 R**=**hit **/** a #Recall

 P**=**hit **/(**M**\***a**)** #Precision

 **if** R**==**0 **or** P**==**0**:**

 F1**=**0

 **else:**

 F1**=(**2**\***P**\***R**)/(**P**+**R**)** #F1 score

 **return** **[**P**,**R**,**F1**]**

一般而言，准确率和召回率存在一定的制约关系，因而需要观测其综合性能最高的值综合判断。根据算法实现得到如下表格图结果：



表2（从上往下M=3,5,7,15,20）基于物品协同过滤（上0-4）、随机推荐（中0-4）、热门推荐（下0-4）评估结果

利用matplotlib绘图库绘制关于准确度、召回率、F1分数（F1 score）分析数值图，如图2所示。



**图2 M=3,5,7,15,20下的P-R折线图**

从表2和图2对比中可以看出，无论是随机算法还是热门推荐算法推荐效果都远不如协同过滤算法效果好，这也说明了协同算法效果的。从表2中观察可以得到，对于协同过滤算法，当M=3时，无论准确度、召回率、F1 score效果都比其他M值时高，因而选择M=3时用作商品推荐效果最佳。

**4 小结**

本案例通过电子商务网站用户的点击行为作为数据分析对象，通过对数据的分析，根据数据的实际情况采用基于内容的协同过滤推荐方法对用户购物行为数据进行建模，并给出不同推荐物品值下的推荐效果。并同时对比了在同等条件下该算法与随机推荐、热门推荐的推荐效果，结果表明，在用户有历史行为记录的情况下推荐效果远优于非个性化推荐算法。

**5附件**

1. 图1数据爬取流程图
2. 图2商品热门指数分布图
3. 图3 P-R折线图
4. 表1：不同模型评估表



**图1数据爬取流程图**

**图2商品热门指数分布图**

**图3 M=3,5,7,15,20下的P-R折线图**



**表1评估结果**

**6 思考题**

1. 当偏好矩阵和相似矩阵过大或过于稀疏时有哪些解决方法？
2. 准确度，召回率，F1 分数分别衡量了数据的什么指标？
3. 如何利用Pearson相关系数、夹角余弦相似度法计算相似度？并用代码实现。
1. **案例编写者**:饶泓、曾雪强、周兴斌、韦勇

**案例涉及的知识点：**推荐系统、数据分析、数据清理、数据挖掘、协同过滤、Python语言及Numpy、Pandas、Sklearn、Matplotlib等库的使用。

**案例真实性：**本组成员郑重声明：所提供的案例数据来源是课题组从互联网上爬取的所得，所有的分析处理工作为本课题组原创，绝无抄袭剽窃等行为。

**案例来源：**本案例通过网络获取国外电子商务网站Retailrocket的用户购物行为脱敏处理数据进行分析处理。 [↑](#footnote-ref-1)