基于Item-CF的电商推荐初探

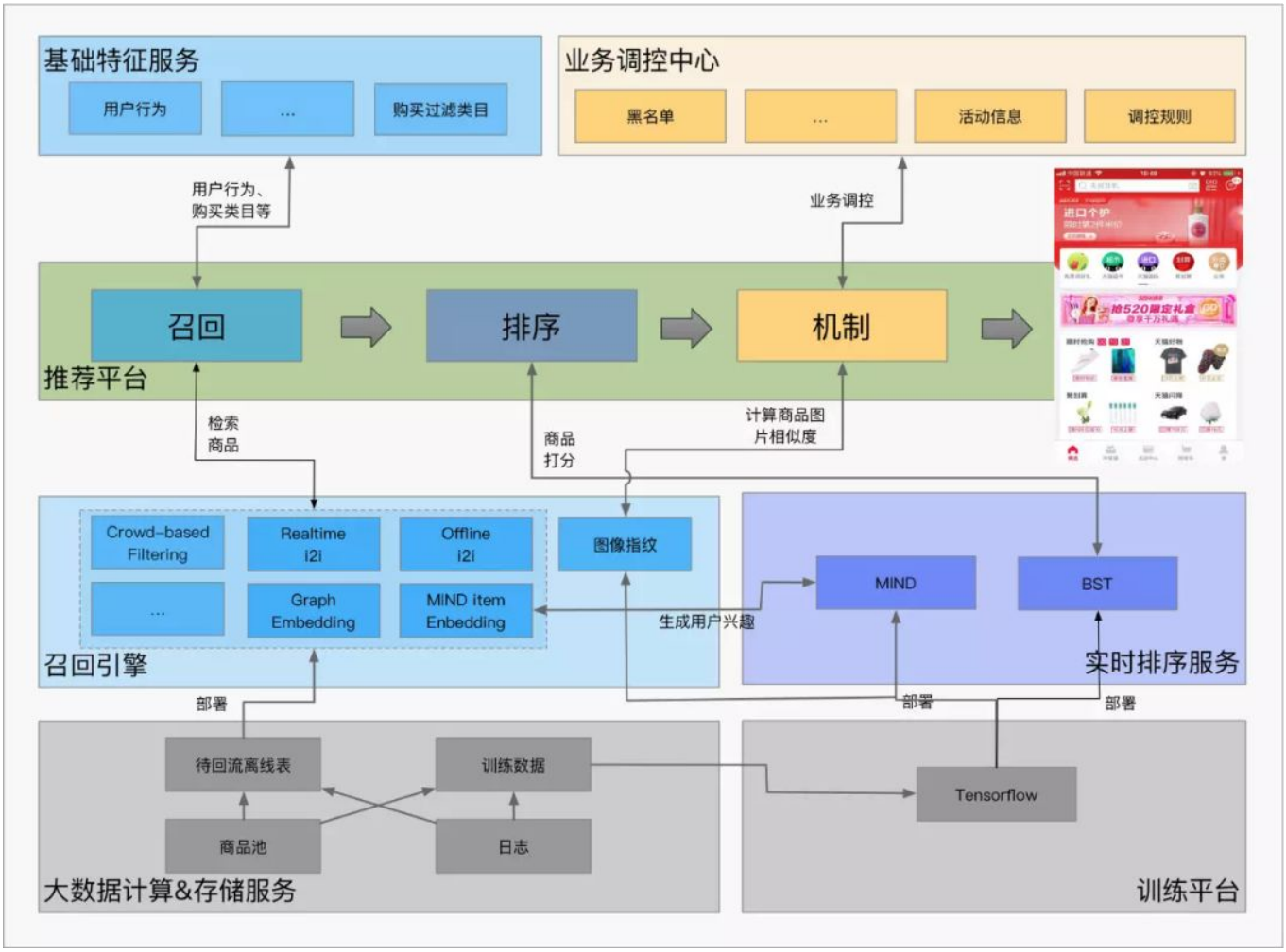
原创 AmyWei AmyWei 2019-06-04

派对动物

五月天 - 自传

00

引言



天猫首页的个性化推荐系统可以分为召回、排序和机制三个模块。

"召回模块主要是从全量的商品素材中检索出用户感兴趣的 TopK 个候选商品。

Item-CF 是目前应用最广泛的召回算法，其原理是根据两个商品被同时点击的频率来计算两个商品之间的相似度 simScore，得到 i2i 表；然后通过用户的 trigger 去查询 i2i 表，扩展用户感兴趣的商品。Item-

CF 的基本算法虽然简单，但是要获得更好的效果，往往需要根据实际的业务场景进行调优。清除爬虫、刷单等噪声数据，合理选择计算商品之间相似度的数据的时间窗口，引入时间衰减，只考虑同一个类目下商品对，归一化、截断、打散等策略对优化 Item-CF 的效果都有很大的帮助。"

看了《当你打开天猫的那一刻，推荐系统做了哪些工作？》，我们知道天猫的推荐系统是相当复杂的。作为小白，咱们只好先从最简单的基于**Item-CF**的推荐算法实践来初探一下电商的推荐。

01

场景分析

协同过滤（Collaborative Filtering：CF）是很常用的一种推荐算法，包括三种类型：

1. 基于用户的协同过滤(User-based CF)，通过计算用户和用户的相似度找到跟用户A相似的用户B, C, D...再把这些用户喜欢的内容推荐给A；

2.基于物品的协同过滤（Item-based CF），通过计算物品和物品的相似度找到跟物品1相似的物品2, 3...再把这些物品推荐给看过物品1的用户们；

3. 基于模型的协同过滤（Model-based CF），主流的方法可以分为：矩阵分解，关联算法，聚类算法，分类算法，回归算法，神经网络。

电商的推荐系统能够很好的发掘物品的长尾，挑战传统的二八原则（80%的销售额来自20%的热门品牌）。当已经对用户行为进行分析得到用户喜好后，可以根据用户喜好计算相似用户和物品，然后基于相似用户或者物品进行推荐。Item-CF 和 User-CF 是基于协同过滤推荐的两个最基本的算法。

User-CF vs Item-CF

User-CF：基于用户对物品的偏好找到相似用户，然后将相似用户喜欢的推荐给当前用户。（**假设用户会喜欢那些和他有相同喜好的用户喜欢的东西**）

Item-CF：基于用户对物品的偏好找到相似的物品，然后根据用户的历史偏好，推荐相似的物品给他。（**假设用户会喜欢和他以前喜欢的东西相似的东西**）

前面介绍了 User-CF 和 Item-CF 的基本原理，对于**电商场景**通常会觉得 Item-CF 从性能和复杂度上比 User-CF 更优**可能**是基于以下几个方面：

1. 对于电商网站，用户的数量远超过物品的数量，同时物品的数据相对稳定，因此计算物品的相似度不但计算量较小，同时也不必频繁更新。
2. Item-CF 的推荐成为了引导用户浏览的重要手段 。（比如在购书网站上，当你看一本书的时候，推荐引擎会给你推荐相关的书籍，这个推荐的重要性远远超过了网站首页对该用户的综合推荐。）
3. 同时Item-CF 便于为推荐做出解释。（在电商的网站中，给某个用户推荐一本书，同时给出的解释是某某和你有相似兴趣的人也看了这本书，这很难让用户信服，因为用户可能根本不认识那个人；但如果解释说是因为这本书和你以前看的某本书相似，用户可能就觉得合理而采纳了此推荐。）

所以单从复杂度的角度，这两个算法在不同的系统中各有优势，推荐引擎的设计者需要根据自己应用的特点选择更加合适的算法。

02

Item-CF

【原理】

给用户推荐那些和他们之前喜欢的物品相似的物品。不过Item-CF不是利用物品的内容计算物品之间相似度，而是利用用户的行为记录（例如，最开头提到的天猫根据两个商品被同时点击的频率来计算两个商品之间的相似度，本文我们将根据用户购买商品行为来计算两个商品之前的相似度）。

从上述概念出发，定义物品i和j的相似度为

$$w_{ij}=\frac{|N(i)\cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

其中，$|N(i)|$是喜欢物品i的用户数，$|N(i)\cap N(j)|$是同时喜欢物品i和物品j的用户数。分母是惩罚物品i和j的权重，因此惩罚了热门物品和很多物品相似的可能性。

在得到物品相似度之后，ItemCF通过以下公式计算用户u对未产生行为的物品j的感兴趣程度。

$$p_{uj}=\sum_{i\in N(u)\cap S(j,K)} w_{ji}r_{ui}$$

这里的$N(u)$是用户喜欢的物品集合，$S(j,K)$是和物品j最相似的K个物品的集合，w_{ij}是物品j和i的相似度，r_{ui}是用户u对物品j的兴趣评分（简单的，如果用户u对物品i有过行为，即可令r_{ui}=1）

【步骤】

1、将物品的用户当作物品的特征向量，然后物品产品之间的相似度，得到物品相似度矩阵。

2、从用户已经产生行为的物品中找到于其相似的K个物品（从相似度矩阵中），两次加权累和（用户产生行为物品的评分*相似物品相似度），找出评分最高的N件物品推荐给用户。

【举例】

1、收集用户数据

例如：用户A购买物品a b d，用户B购买物品b c e，用户C购买物品c d，用户D购买物品b c d，用户E购买物品a d。

2、找出物品相似度

```
物品被多少个不同用户购买：
N: {'a': 2, 'b': 3, 'd': 4, 'c': 3, 'e': 1}
物品的共现矩阵：
C: {'a': {'b': 1, 'd': 2}, 'b': {'a': 1, 'd': 2, 'c': 2, 'e': 1}, 'd': {'a': 2, 'b': 2, 'c': 2}, 'c': {'b': 2, 'e': 1, 'd': 2}, 'e': {'b': 1, 'c': 1}}
```

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

转换公式即：

```
self.W[i][j] = cij / (math.sqrt(N[i] * N[j]))  # 按上述物品相似度公式计算相似度
```

结果为：

```
物品相似度矩阵：
a:{'b': 0.4082482904638631, 'd': 0.7071067811865475}
b:{'a': 0.4082482904638631, 'd': 0.5773502691896258, 'c': 0.6666666666666666, 'e': 0.5773502691896258}
d:{'a': 0.7071067811865475, 'b': 0.5773502691896258, 'c': 0.5773502691896258}
c:{'b': 0.6666666666666666, 'e': 0.5773502691896258, 'd': 0.5773502691896258}
e:{'b': 0.5773502691896258, 'c': 0.5773502691896258}
```

用户A购买过的商品（a b c）的相似度矩阵

```
物品相似度矩阵：
a:{'b': 0.4082482904638631, 'd': 0.7071067811865475}
b:{'a': 0.4082482904638631, 'd': 0.5773502691896258, 'c': 0.6666666666666666, 'e': 0.5773502691896258}
d:{'a': 0.7071067811865475, 'b': 0.5773502691896258, 'c': 0.5773502691896258}
```

进行K=3 排序

a:{'d': 0.7071067811865475, 'b': 0.4082482904638631}
b:{'c': 0.6666666666666666, 'd': 0.5773502691896258, 'e': 0.5773502691896258}
d:{'a': 0.7071067811865475, 'b': 0.5773502691896258, 'c': 0.5773502691896258}

3、找出评分最高的N=10件物品推荐给用户

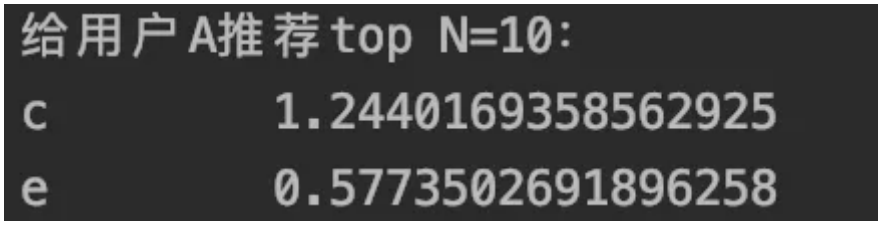
$$p_{uj} = \sum_{i \in N(u) \cap S(j,K)} w_{ji} r_{ui}$$

去掉用户已购买的a b d，计算c 和 d：

c: 0.66+ 0.57 = 1.23

e:0.57

排序取top N=10 推荐给用户:

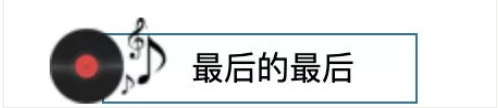


【代码】

```
import math

class ItemBasedCF:
    def __init__(self,train_file):
        self.train_file = train_file
        self.readData()

    # 读取数据
    def readData(self):
        self.train = dict()

        for line in self.train_file:
            user,score,item = line.strip().split(",")
            self.train.setdefault(user,{})
            self.train[user][item] = int(float(score))

        # 输出数据集-用户(兴趣程度，物品)
        print ("输出测试数据集合：")
        print (self.train)


    # 物品相似度矩阵
    def ItemSimilarity(self):
        C = dict()  # 物品-物品的共现矩阵
        N = dict()  # 物品被多少个不同用户购买
        for user,items in self.train.items():
            for i in items.keys():
                N.setdefault(i,0)
                N[i] += 1  # 物品i出现一次就计数加一
                C.setdefault(i,{})
                for j in items.keys():
                    if i == j : continue
                    C[i].setdefault(j,0)
                    C[i][j] += 1  # 物品i和j共现一次就计数加一
        print("物品被不同用户购买次数：")
        print ('N:',N)
        print("物品的共现矩阵：")
```

```
        print ('C:',C)

        # 计算相似度矩阵
        self.W = dict()
        for i,related_items in C.items():
            self.W.setdefault(i,{})
            for j,cij in related_items.items():
                self.W[i][j] = cij / (math.sqrt(N[i] * N[j]))  # 按上述物品相似度公式计算相似度
        print("物品相似度矩阵：")
        for k,v in self.W.items():
            print (k+':'+str(v))
        return self.W

    # 给用户推荐前N个最感兴趣的物品
    def Recommend(self,user,K=3,N=10):
        rank = dict() # 记录用户的推荐物品（没有历史行为的物品）和兴趣程度
        action_item = self.train[user]     # 用户购买的物品和兴趣评分r_ui
        for item,score in action_item.items():

            for j,wj in sorted(self.W[item].items(),key=lambda x:x[1],reverse=True)[0:K]:  # 使用与物品item最相似的K
                if j in action_item.keys():  # 如果物品j已经购买过，则不进行推荐
                    continue
                rank.setdefault(j,0)
                rank[j] += score * wj  # 如果物品j没有购买过，则累计物品j与item的相似度*兴趣评分，作为user对物品j的兴
        return dict(sorted(rank.items(),key=lambda x:x[1],reverse=True)[0:N])

# 测试集
uid_score_bid = ['A,1,a','A,1,b','A,1,d','B,1,b','B,1,c','B,1,e','C,1,c','C,1,d','D,1,b','D,1,c','D,1,d','E,1,a','E,1,d']

# 声明一个ItemBased推荐的对象
Item = ItemBasedCF(uid_score_bid)

# 商品相似度矩阵
Item.ItemSimilarity()

# 计算给用户A的推荐列表-K=3,N=10
print("给用户A推荐（K=3,N=10）：")
recommedDic = Item.Recommend("A")
for k,v in recommedDic.items():
    print (k,"\t",v  )
```

代码原型来源于网上，用以帮助加深对Item-CF算法的理解



这篇文章只是一次简易的通过Item-CF算法实践来初探电商推荐系统的学习笔记，推荐系统比较复杂，还需要不断学习。