LookAlike

2015年11月23日 17:00

表

r_auction_auctions
tc_biz_order

order_days.sh
产生店铺种子数据

提取多天购买数据(90天内)
item_id ^A buyer_id ^A cat_id ^A buyer_nick ^A seller_id ^A create_time ^A seller_nick

提取店铺种子数据
seller_nick ^A user_id
在seller_nick店铺内有过购买行为的用户

data.sh
基础数据处理

提取购买数据，类目过滤
item_id ^A buyer_id ^A cat_id ^A buyer_nick ^A seller_id ^A create_time

提取收藏数据 (创建时间在day_num天之内)
user_id ^A user_nick ^A item_id ^A serller_id ^A item_type ^A create_time

提取浏览数据
item_id ^A user_id ^A user_nick ^A seller_id ^A visit_time

item_property.sh
获取item属性

从宝贝表里提取属性数据
key：auction_id(18)
value：title(6) \t ws_title \t category_name(1) \t price(5) \t property_list
ws_title是对title的分词处理
property_list的格式：property_name:property_value_name ^A

preprocess.sh
处理店铺特征

提取店铺特征
user_id ^A seller_id ^A score

score的计算逻辑如下：
如果是店铺收藏或宝贝收藏，加1；
如果有购买，加1；
如果有点击，加1。
总分就是user_id和seller_id之间的分数

统计店铺特征分布
seller_id的记录数cnt

店铺特征过滤
获取每个seller_id的cnt，过滤掉小于1000的记录
user_id ^A seller_id ^A score

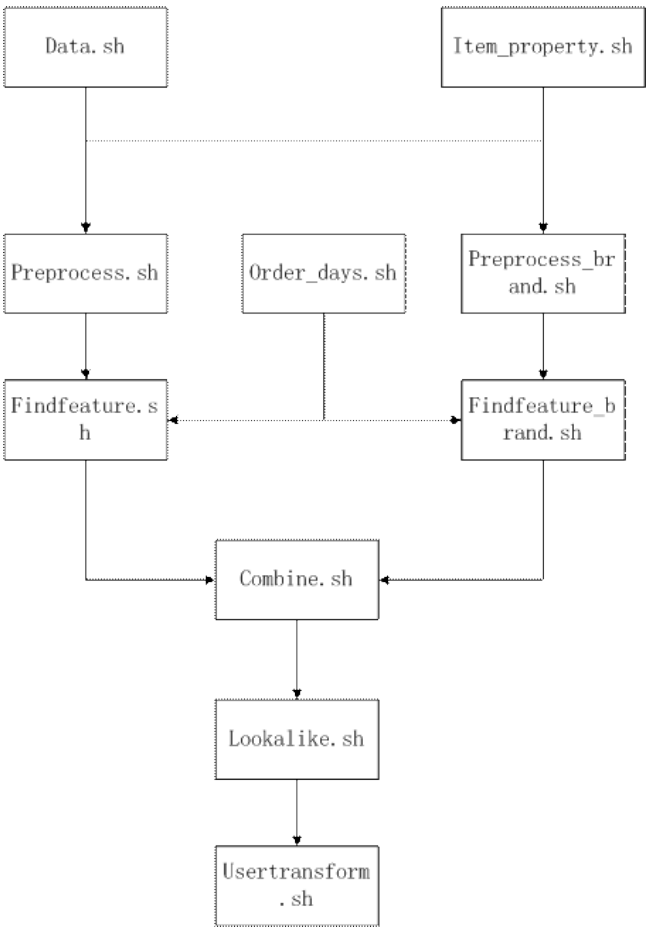
preprocess_brand.sh
处理品牌特征

提取品牌特征 (要看代码)
user_id ^A item_id ^A brand ^A 1

1. 购买
2. 点击
3. 收藏

合并品牌求和
user_id ^A brand ^A total_score
对user_id计算其在每个brand上的得分（求和）

统计品牌特征分布



提取在店铺内有过购买行为的用户user_id

feature：

用户还访问过的店铺的特征

1. seller_total_cnt: 在该店铺的有购买收藏点击行为的人数
2. count: 购买+点击+收藏
3. cnt: 根据【相关的店铺总数】【在该店铺有行为的用户数】【相关用户总数】计算出

用户还买过的品牌的特征

1. feature_total_cnt: 在该品牌有购买点击收藏行为的人数
2. count: 购买+点击+收藏
3. cnt:根据【相关的品牌总数】【在该品牌有行为的用户数】【相关用户总数】计算出

seller_nick\tfeature:score:count:number:temp_maxN\tfeature:score:count:number:temp_maxN \t
cnt作为特征的score

店铺的特征:
种子用户还喜欢的店铺和品牌的分数

店铺---
特征：与该店铺相关的店铺或品牌
分数：相关程度

用户的特征：
用户对店铺或品牌的喜爱程度

用户---
分数：对店铺或品牌的喜欢程度

lookalike过程

对于店铺的每个特征
计算 用户---特征 的分数

user_id ^A brand ^A total_score
对user_id计算其在每个brand上的得分（求和）

统计品牌特征分布
brand ^A cnt
统计每个brand的记录数cnt

findfeature.sh
产生店铺特征及计算

关联店铺特征信息
user_id ^A seller_nick ^A seller_id ^A score
用户---种子店铺---店铺---分数

关联店铺整体分布信息
user_id ^A seller_nick ^A seller_id ^A score ^A total_cnt

计算特征分数
seller_nick \t seller_id:cnt:count:seller_total_cnt \t seller_id:cnt:count:seller_total_cnt \t...
与每个seller_id对应的cnt,count,seller_total_cnt的计算逻辑：
    count=sum(score)
    seller_total_cnt = total_cnt

cnt的计算逻辑复杂一点，分以下几步进行：
pxy = count/不同seller_id的个数
py= feature_total_cnt/70000000
cnt=count/不同user_id的个数
cnt = (float)Math.pow(cnt /不同user_id的个数, 0.5) * (pxy / (py + 0.00001f));
cnt = count * ratio

findfeature_brand.sh
产生品牌特征及计算

关联品牌特征信息
user_id ^A seller_nick ^A brand ^A score
用户--种子店铺---品牌---分数

关联品牌整体分布信息
user_id ^A seller_nick ^A brand ^A score ^A total_cnt

计算特征分数
seller_nick \tbrand:cnt:count:feature_total_cnt \t brand:cnt:count:feature_total_cnt \t ...
特征的分数score就是cnt

与每个brand对应的cnt,count,feature_total_cnt的计算逻辑：
count=sum(score)
feature_total_cnt = total_cnt
cnt的计算逻辑复杂一点，分以下几步进行：
pxy = count/不同brand的个数
py= feature_total_cnt/70000000
cnt=count/不同user_id的个数
cnt = (float)Math.pow(cnt /不同user_id的个数, 0.5) * (pxy / (py + 0.00001f));
cnt = cnt * ratio

combine.sh
特征数据合并

seller_nick\tfeature:score:count:number:temp_maxN\tfeature:score:count:number:temp_maxN \t ....

brand/seller_id:cnt:count:feature_total_cnt
特征的分数score就是cnt

1. number=sum(count); temp_maxN=maxN*2.5; (maxN是一配置参数，取350000)
2. 将feature按score值由大到小排，如果前n个feature 的number值大于等于temp_maxN，则停止拼key，输出结果。

lookalike.sh
lookalike过程
(要看代码)

计算店铺和用户的分数
seller_nick^Auser_id^Ascore

店铺的分数feature_score

lookalike过程

对于店铺的每个特征
计算 用户---特征 的分数
score= feature_score * log(1+ user_feature_score)

计算用户到店铺的总分
seller_nick^Auser_id^Atotal_score^Acount

去掉种子用户

对每个seller_nick 的所有user_id，按score值从大到小排序

来自feature
店铺和品牌的分数

用户的分数 user_feature_score
用户对店铺对品牌的分数
user_id ^A seller_id ^A score
user_id ^A brand ^A total_score

score= feature_score * log(1+ user_feature_score)

分数分组求和
seller_nick^Auser_id^Atotal_score^Acount
1）total_score为map输出的score加和; count为输入reduce的key对应的记录条数
2）total_score如果小于score_threshold，则过滤掉

去掉种子用户
如果user_id是店铺seller_nick的种子用户，则过滤掉该记录

排序
对每个seller_nick 的所有user_id，按score值从大到小排序