推荐系统

算法工程师负责，推荐系统

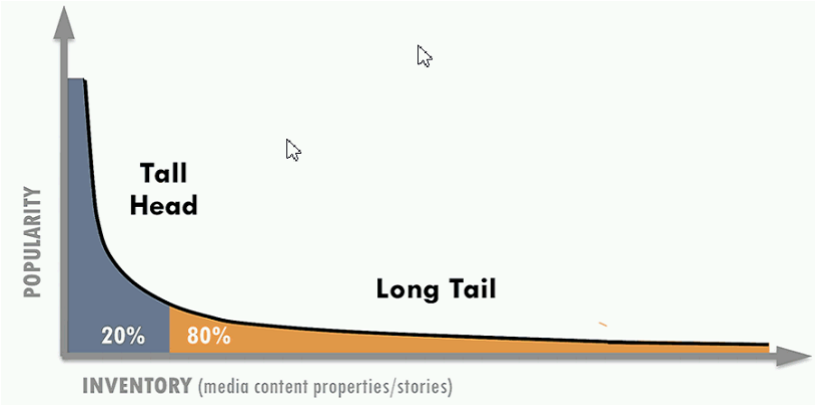
根据场景构建自己的推荐系统

由来

用于解决信息过载（

- 信息量大
- 解决：漏斗减少最终呈现给用户的信息数量（N -> 1）

）和长尾效应（

- 二八原则
- 

    - tall head：基于热度的推荐（topN）
    - long tail：个性化推荐

）

> 主观能动性：
>
> - 没有对错，只有行不行和好不好，不好就优化。
> - 自己分析自己想，先做，然后解决问题

推荐item

都是item，但是具体形态各不相同

推荐要有方法和侧重点（因材施教）

item复杂 -> 融合推荐

考虑用户时长

特征构建（

- 文本相似性

）

> 数据量：

> 配置：
>
> 服务器：
>
> 运行时长：几十分钟到几个小时（数据量，处理，引擎）
>
> 一条数据：1k 100k~1m

> 回答不能模棱两可，要明确，坚决

优化的时候评估数据量和任务数

> 小公司三面
>
> - 一面二面人事
> - 技术面 一两轮
> - 人事 价值观
> - 领导的领导 聊天

上下游

> 技术面
>
> - 大部分根据自己的简历

算法

协同过滤（CF）

- 基于用户（用户之间的相似性）【多用】【U2I】
- 基于商品（商品之间的相似性）【u2i】【可以U2I2I】
    - 冷启动问题
    - 误操作
- 热度
- 内容
    - 标签化（标签预测，标签分类）
- 关联规则
    - 两个商品单独出现少，两个一起出现多。
    - 避免热门
- 模型
- 混合

> - 推荐场景有足够的的认识
>   - 特点
>   - 侧重点
> - 整体流程架构，因果关系，流程和逻辑

架构

亚马逊：离线（

- 训练模型
- 得到模型打分
- 输出item List

）、近线（

- 存储
- 召回
- 离线算的

）、在线（

- 收集用户行为
- 在线结果
- 提供实时特征
- 推荐的web服务（商品、排序（近线取的））

）

模型预测（离线和在线结合）

一般是二分类模型（收集正负例样本）

- 反馈：用户喜欢和不喜欢
- 程度

架构拆解

推荐web服务：

四步骤：

- 召回（数据太多，挑出用户可能感兴趣的item，根据u2i或者i2i等算法，冷启动用热门兜底）matcher
    - 硬件性能：计算量太庞大
    - 推荐候选集
    - 获取特征（
        - 用户特征
        - 商品特征
        - 用户和商品交叉特征

      ）
    - 输出：离线/实时 存到近线服务
- 过滤 filter
    - 补全特征信息
    - 过滤一些实时信息（黄赌毒，不合规范，售完等）
- 打分/排序（粗排）ranker
    - 入：样本和样本特征
    - 出：预测打分（0~1区间的值）
- 排序/重排（精排）reranker

    - 对粗排进行调整
        - 加权减权
        - banner置顶
        - 曝光指定商品
        - 业务规则调整在这个阶段
        - 各种推荐的组合规则（组合推荐商品、文章、店铺等）

近线服务

数据库

- 小 关系型
- 大 非关系

模型在线服务

- 在线做预测
- 模型文件：一堆特征和权重值
- 一般单独服务（解耦），可以放到推荐web服务中
    - 放进去不利于更新迭代
    - 内存交换问题，更新不影响用户发送请求进来及处理

缓存

- 防止恶意刷新
- 并发性考量

实时任务

实时特征计算

实时数据召回

离线任务

离线数据清洗

- 特征操作/工程

离线模型训练

- 召回类算法
- 打分类算法
- 其他

> 特征工程为构建模型服务。
>
> 取出有用模型
>
> - 归一化处理（梯度下降法求解的模型（线性回归，逻辑回归，支持向量机，神经网络等模型））

- why
  - 消除数据特征之间的量纲化（比如距离单位米和重量单位千克，如果要分析一个人体重和身高对健康的影响，因为身高特征在1.6和1.9m之间，体重在50~100kg之间，那么结果会偏向数值差别较大的体重）影响。
- 方法
  - 线性函数归一化
  - 零均值归一化
- 不适合决策树模型（信息增益比）

实际操作：防止最大值和最小值相同而报错，所以加一个很小的数

```
=dowFeature::getMinMicOnStay).mapToDouble(e -> ((float)e - min2) / (max2 - min2+ 1e-5)).toArray();
```

名词

ctr 点击率

cvr 购买转化率

问题

ctr&cvr（数量级的不公平，修正指标【威尔逊算法】）

用户体验和用户粘性（留存率）

提高人均充值消费金额

面试

面技术

面业务（重要）

基于场景思考

思考问题和解决问题

关键项目要能捋清楚

标签体系

用户画像

标签构建，分析

召回

兜底

一般是组合召回，可以开多线程进行多路召回

不是一个预测，得到的结果是数据信息，比如item

优质召回

实时数据

而模型预测是输入特征，user特征，item特征，交叉特征等，输出预测itemList

预测代码

先看结构/架构，然后业务，然后细节

入口：recommendController

封装上下文

（用户交互问题对房间标签的准确性：以前依赖房主设定，改进为房间人数定期投票（以多为主））

（提供评估）分流：AB分桶分流

通过上下文的信息计算用户的策略分桶（hash把用户分到哪个桶），返回四步的实现类信息

通过这个实现类信息可以获得此用户进行的具体四步用的算法

可以对推荐模型进行对比差异

```
base-A:  0~199
match:NewUserQualityMatchingIMPL
filter:RoomFeatureFilterIMPL
ranker:OnlineModelRanker
reranker:DeadRuleReranker
base-B:  200~399
match:NewUserQualityMatchingIMPL
filter:RoomFeatureFilterIMPL
ranker:OnlineModelRanker
reranker:DeadRuleReranker


base-C: 400~999
match:NewUserQualityMatchingIMPL
filter:RoomFeatureFilterIMPL
ranker:OnlineModelTORRanker 
reranker:MatchLevelBaseRuleReranker


```

AA分桶探究波动性

AB分桶探究对比优化

springBean映射的方式实现方法。

matcher

召回级别名称

多线程框架transmittable-thread-local（阿里开源）

RoomFeatureGen

威尔逊区间算法

flink统计一段时间的信息，归一化，加权加一起，得到优质房间的分数。

item相似度找用户群体相似度，基于用户。

node2vec

独热编码

```
onehot编码
样本是一些人和人的基本信息...预测是穷（年收入百万以下）还是富（年收入百万以上）
年龄..地域..学历...公司...工作年限..从事行业
userA.......本科.1
userB→硕士→2
userC→博士→3
userD....高中→0


学历..1本科..2.硕士...3博士..0高中
.
....|....是否高中..是否本科..是否硕士..是否博士
userA...0→→→1→→→0→→→0
userB→0→→→0→→→1→→→0
userC→0→→→0→→→0→→→1
userD...


room001....userA..userB..userC
room002→→userB.userC..userD
room003→→userD.userE

userA.→→0
userB→→1
userC→→2
userD→→3
userE→→4


room001→→1→→1→→1→→0→→0
room002→→0→→1→→1→→1→→0
room003→→0→→0→→0→→1→→1
```

onehot编码

```
room001 userA userB userC
room002 userB userC userD
room003 userD userE

room001 0
room002 1
room003 2

userA 1 0 0
userB 1 1 0
userC 1 1 0
userD 0 1 1
userE 0 0 1
```