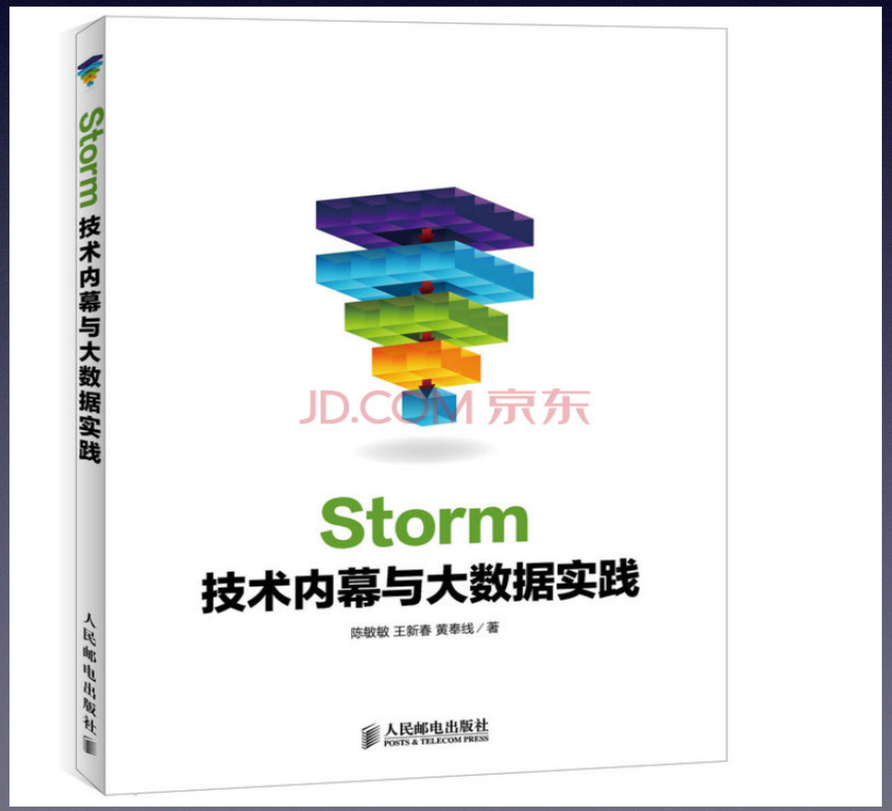
电商场景下的千人千面与实时意图实践

陈敏敏

自我介绍

从事搜索、推荐、大数据平台相关工作，目前主要关注实时计算框架、推荐系统、大数据营销、技术架构。

书：

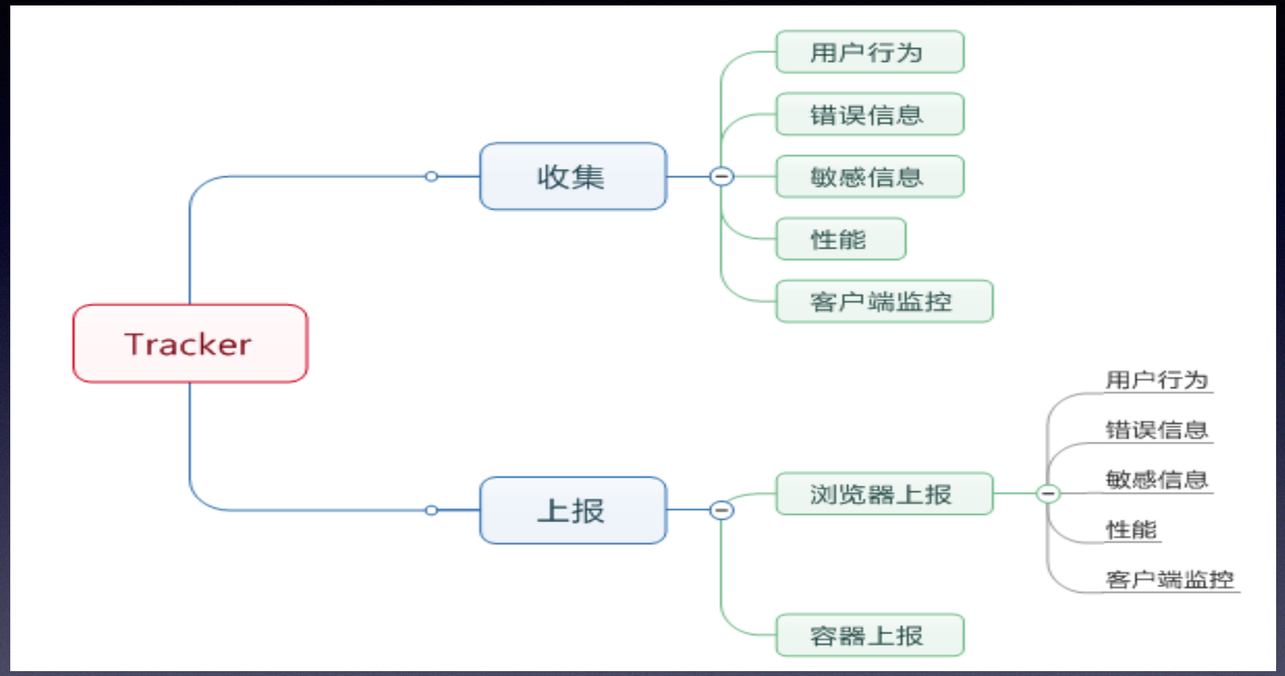


QQ：24180823

主要内容

- 推荐平台之业务架构
- 实时意图
- 意图之基础架构
- 意图之应用架构

数据=》算法=》业务



结合其他数据：
竞争对手数据(爬虫)、
商品信息、用户信息…

机器学习、
NLP等算法

推荐
广告投放
个性化搜索
选品
BI报表
供应链优化
智能定价
精准化营销
优化购物路径
…

推荐产品业务架构

转化率 交叉销售

个性化推荐

推什么？

全面深挖用户购物兴趣

= 短期兴趣 转化率 + 长期累积兴趣 转化率 交叉销售 + 潜在兴趣 交叉销售

短期兴趣 转化率	长期累积兴趣 转化率 交叉销售	潜在兴趣 交叉销售
用户意图引擎	用户画像引擎	千人千面引擎 / CF推荐引擎
高转化率 精准定位用户短期兴趣 用于首页栏位、站外广告 EDM及APP营销投放	**长期累积的兴趣图谱** 1亿userID, 5亿GUID 每个用户的类目、品牌、导购属性兴趣偏好	**用户群体兴趣** 覆盖8千万左右用户 购买力level、同学(985,文科) 男/女、地域、同事、邻居、 同行、好友群、辣妈、孕妇、 新/老客 **相似用户兴趣** 协同过滤算法挖掘 相似用户兴趣

什么时间推？

精准定位购物兴趣发生的时机

= 合适的场景 交叉销售 + 购物周期 转换率 + 最近热点 交叉销售

合适的场景 交叉销售	购物周期 转换率	最近热点 交叉销售
情境推荐引擎	反向推荐引擎	主题推荐
定位特定类目购物情境 公历农历的季度、月和周、 中西节日、当地气温变化、当 地天气、外出、旅游	**定位复购时间点** 定位复购时间点 覆盖74个高复购率类目 将用户划分成新客、成长期、衰 退期、流失期多个阶段，投放不 同的类目及负毛利营销品	**用户行为兴趣** 根据评论、标题给商品和用户 打标签形成场景词 形成主题场景聚合SKU

拉升销量和用户转化率 拉升交叉销售GMV 提升精准化营销效率

各种栏位场景适用推荐算法

	用户场景	业务意图	算法							
			短期意图	长期画像	协同过滤	关联规则	周期购算法	刚需爆款	冲动性爆款	主题推荐
首页	弱目的性闲逛	trade cross		✓	✓	✓	✓	✓	✓	✓
	强目的寻找相关促销或爆款	trade in	✓							
类目/搜索	寻找更适合自己的商品	trade in		✓						✓
	了解同类人群的购买选择	trade in			✓					
详情页	挑选、比较的需求	trade in		✓	✓					
	了解同类人群的购买选择	trade in			✓					
	经济节约的诉求	trade in								
	对相关商品的潜在购买提醒	trade cross				✓				
继续逛页	对相关商品的潜在购买提醒	trade cross				✓				✓
购物车	凑单免邮的需求	trade up		✓		✓	✓	✓	✓	
	对相关商品的潜在购买提醒	trade cross		✓		✓	✓	✓	✓	
	占便宜	trade up	✓	✓		✓	✓	✓	✓	
订单完成页	一次购物周期结束，顺便看看有没有其他购物心动点	trade cross	✓	✓		✓	✓	✓	✓	
消息触达	对于关注的商品、品牌，了解其有利的动态	trade in	✓	✓						✓

推荐后台系统

显示栏位配置

页面ID	页面名称	栏位ID	栏位名称	栏位推荐品数量	栏位状态	操作
4	H5产品详情页	53	测试栏位二排1	156	可用	
1	一号店产品详情页	54	验证栏位和算法修改	20	可用	

显示流程配置

流程ID	流程名称	推荐方式	是否团购	是否闪购	是否过滤库存	类目打散方式	品牌打散方式	推出商品数量	流程可用状态	操作
58	测试用户画像修改	用户画像	否	否	是	无	无	12	可用	
59	测试选品池	带故事情景的纯选品池	是	是	是	交替排序	随机打散	11	可用	
60	测试看了还看	相似相关算法库	否	否	是	随机打散	交替排序	10	可用	

效果预览

当前栏位 验证栏位和算法修改

当前算法名称 测试用户画像修改　　当前算法优先级 2

当前算法流量控制 0 至 0　　当前算法推出商品数 12

当前算法名称 测试选品池　　当前算法优先级 2

千人千面

天气维度
换季、气温、雨雪、雾霾
【覆盖全国2954个市、县、区】

节日维度
农历节日：春节、端午、中秋、节气等
西历节日：元旦、国庆、父亲节、母亲节
大促：双11，双12，12.21，店庆
【覆盖全年共50种各类节日】

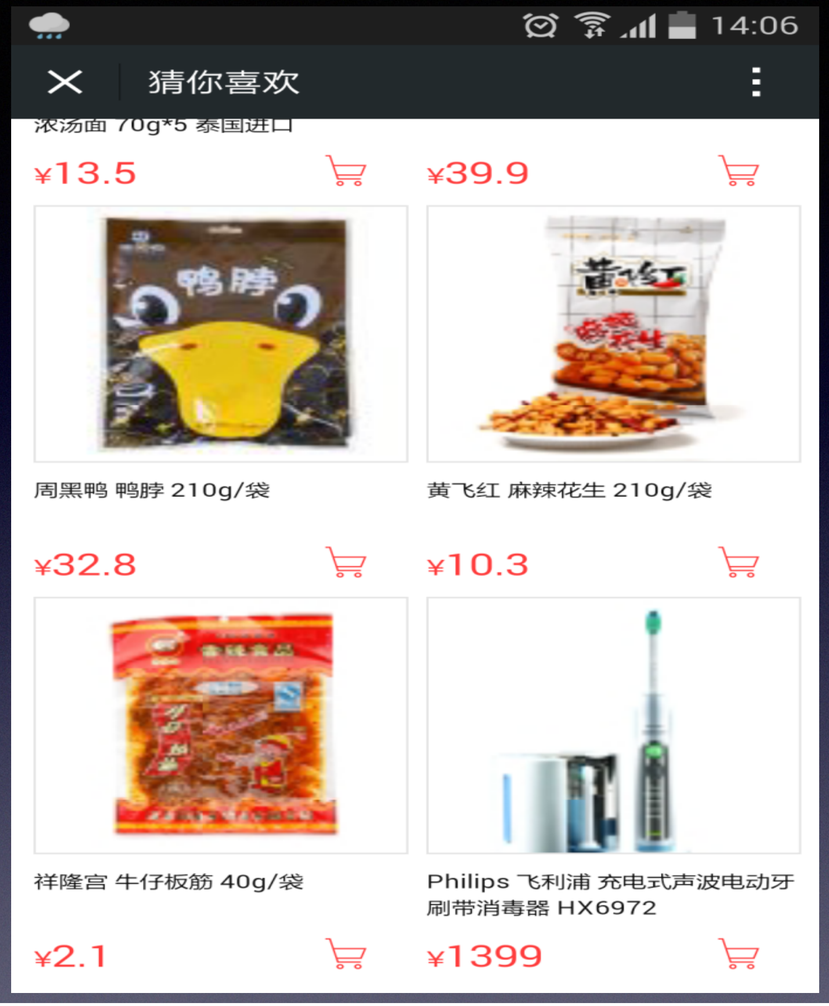
地域维度
大区：东北、华中、华东、华南等
旅游地、城市级别、小区，公司，
小区档次，学校类型，公司类型
【覆盖全国378个地级市或区】

时间维度
月份、季节、星期
【覆盖全年】

画像维度
性别，辣妈，促销敏感

产品数据
DXX，团\闪购，一贵就赔等

前端：



栏位推出物品类型 商品

栏位允许挂载的推荐引擎
- [ ] 用户画像
- [ ] 用户意图
- [x] 纯选品池
- [ ] 带故事情景的纯选品池
- [x] 相似相关算法库

栏位入参种类
- [x] 栏位ID
- [ ] 用户ID
- [ ] 设备ID
- [x] 省份ID
- [x] 主品ID
- [x] 商家ID
- [x] 推荐商品数量
- [x] 推荐商品图片尺寸

基于大数据的千人千面引擎（潜在兴趣）

用户群体千人千面

校园标签【244万校园用户
按校园收货地址、IP、GPS聚合，形成同学群
含985/211、综合/师范/理工/文科/医药/语言

亲朋好友群
按用户手机通讯录聚合客户的亲朋好友及同事，形成社交网络图

公司标签/同事群
按单位收货地址的工作单位聚合，形成同事群
行业、公司规模、收入档次

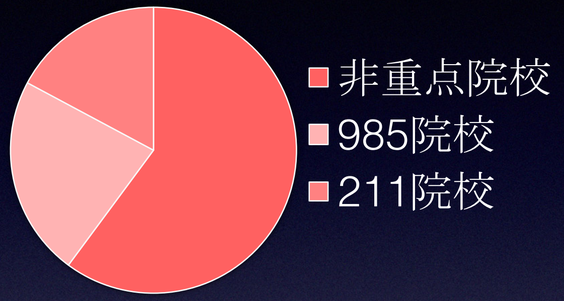
小区标签/邻里群
按小区收货地址及坐标的聚合，形成邻居群
小区名、楼盘档次

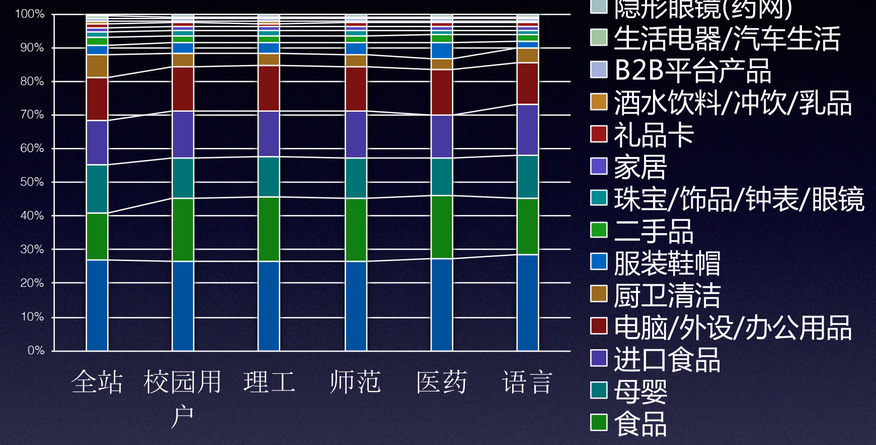
活动地点标签
关注重点购物中心、旅游地，推荐土特产，聚恒隆女，沃尔玛超市附近

辣妈群、性别、购买力等标签
用户画像基础标签

校园、小区、公司用户标签

一期覆盖244万校园用户，985用户55万，211院校41万，其它147万





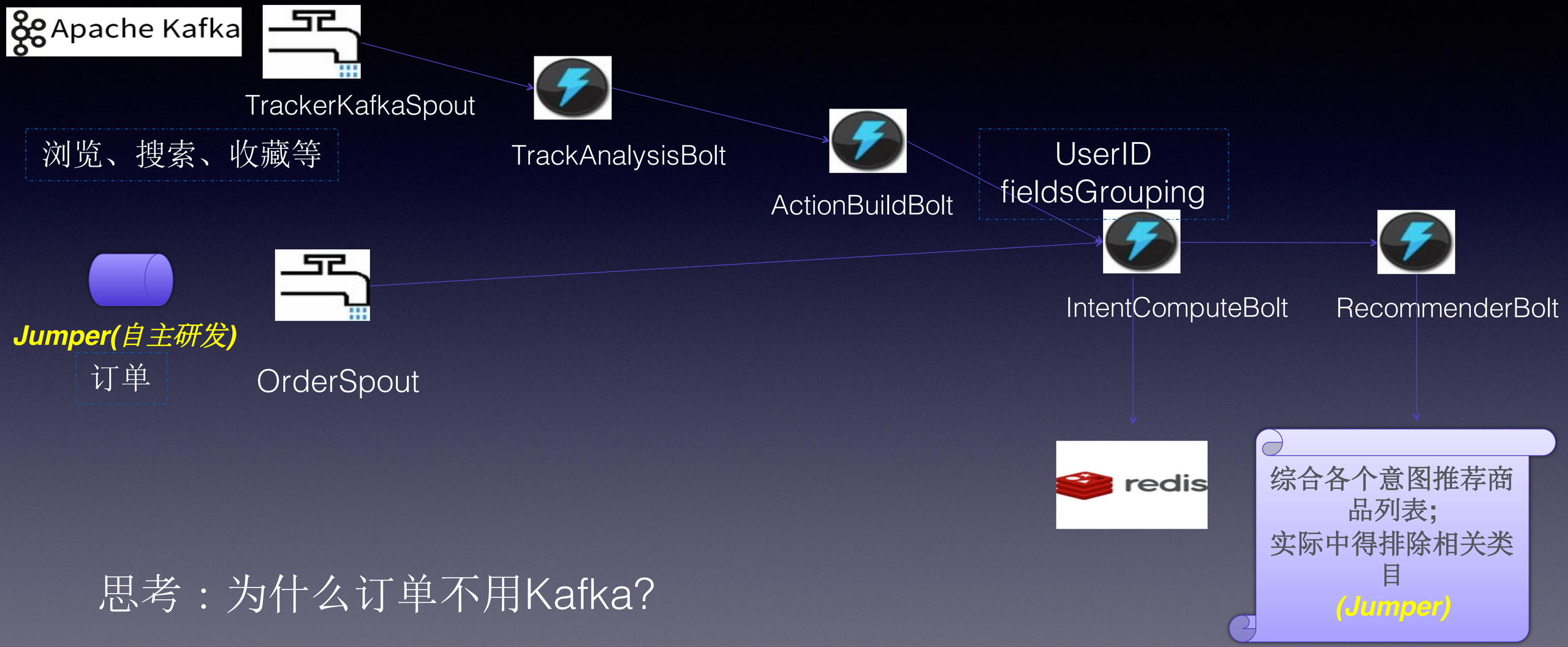
可以换成其它千人千面，如同城、同事、小区、同城孕妇

1号店详情页

主要内容

- 推荐平台之业务架构
- 实时意图
- 意图之基础架构
- 意图之应用架构

实时用户意图：Storm Topology



思考：为什么订单不用Kafka?

意图计算过程

加入购物车	完成了一半的意图
购物车删除	未完成的意图
支付	完成的意图
浏览、搜索、导航、收藏	• 判断意图是否完成，如果是，直接返回； • 获取完成了一半和未完成状态的意图，如果空，设置当前意图到未完成意图； • 更新所有未完成意图，更新得分（遗忘因子衰减，如下）； • 更新所有半完成意图，更新得分（遗忘因子衰减，如下）； • 设置当前意图到最近意图列表中。

$$X_{i+1} = \begin{cases} (1-f)\cdot X_i + x_{i+1} & (x_{i+1} \epsilon I_{now}) \quad \text{行为属于当前行为} \\ X_i + x_{i+1} (x_{i+1} \epsilon I_{now} \cap x_{i+1} = I_{last}) & \text{行为属于当前行为且和前一个行为相等} \\ (1-f)\cdot X_i + 0 & (x_{i+1} \neq I_{now}) \quad \text{行为不等于当前行为} \end{cases}$$

$$\text{遗忘因子 f} = k_1(1-w) + k_2\frac{t}{T} \qquad \text{w} = \frac{1}{1+e^{\alpha-\beta X}}$$

主要内容

- 推荐平台之业务架构
- 实时意图
- 意图之基础架构
- 意图之应用架构

消息消费可靠性回顾

数据传递形式	描述
最少一次（At-least-once）	消息可能会再次发送（没有丢失的情况，但是会产生冗余）。
最多一次（At-most-once）	忍受消息丢失的情况发生
恰好一次（Exactly-once）	每条消息都被发送过一次且仅仅一次（没有丢失，没有冗余）。实现代价比较高

实时框架比较

实时相关的框架	消息传递形式
Storm	At Least Once; Exactly-Once with Trident
Spark	Exactly Once
Kafka	默认保证At least once，允许通过设置producer异步提交来实现At most once
Samza（基于Kafka）	At Least Once
Jumper（1号店自主研发）	At Least Once（自行实现消息的幂等性），broker自动应答； At-most-once，客户端应答

自主研发Jumper

Kafka	Jumper
完全手动，靠各个客户端控制，zookeeper中记录客户端状态	自主决策，服务端保存状态，统一处理和控制
producer使用push模式； consumer使用pull模式；	consumer使用push模型，及consumer server拉取后推送给consumer client
依赖文件系统去存储和cache消息	用mongodb做消息堆积，方便查询历史；
可维护性水平达不到	可维护性强
消息在某些条件下存在丢失的可能	保证消息不丢，但是不保证消息的全局生成和消费顺序

思考：Storm spout是pull模型取消息，那么如何读push模式的消息？

推拉模式的适应场景不同，其中jumper的推模式实时性高，订单对意图变化最明显，用jumper；加车，浏览相对没那么实时，用Kafka；并且通过Jumper把实时推荐的商品列表推送到前端

主要内容

- 推荐平台之产品架构
- 实时意图
- 意图之基础架构
- 意图之应用架构

实时意图中的应用架构举例

屏蔽对象集合的容器类的实现细节，而能对容器内包含的对象元素按顺序进行有效的遍历访问，需要识别每一个行为是加车、搜索、浏览等，适用迭代器模式：

实现Iterable接口生成TrackerAnalyzers：

```
TrackerAnalyzers implements Iterable<IActionAnalyzer>,IActionAnalyzer{
  public Convertable analyze(Tracker tracker) {
    while( iterator().hasNext() && convertable==null ){
       convertable = iterator.next().analyze(tracker);
    }
  }
}
```

Storm中的execute类

```
TrackerAnalyzers  trackerAnalyzer = new TrackerAnalyzers();
trackerAnalyzer.addAnalyzer(new WLAddCartActionAnalyzer());
trackerAnalyzer.addAnalyzer(new WLSearchActionAnalyzer());
…
Convertable actnConvertable = trackerAnalyzer.analyze(tracker);
```

THANKS !