大数据用户画像设计与实践

主讲人：李希沅，姚劲

奈学教育

01 二手电商用户画像实践

01 什么是用户画像

信息 -> 文本 -> 标签

用户画像(User Profile)

用户信息标签化

通过收集用户的社会属性，消费习惯，偏好特征等各个维度的数据，进行对用户或者产品特征属性进行刻画，并对这些特征进行分析，统计，挖掘潜在价值信息，从而抽象出用户的信息全貌

02 用户画像的作用

业务决策

统计指标展示

排名统计

地域分析

行业趋势

竞品分析

精准营销

信息定点推送

邮件

短信

APP推送

个性化服务

个性化搜索推荐

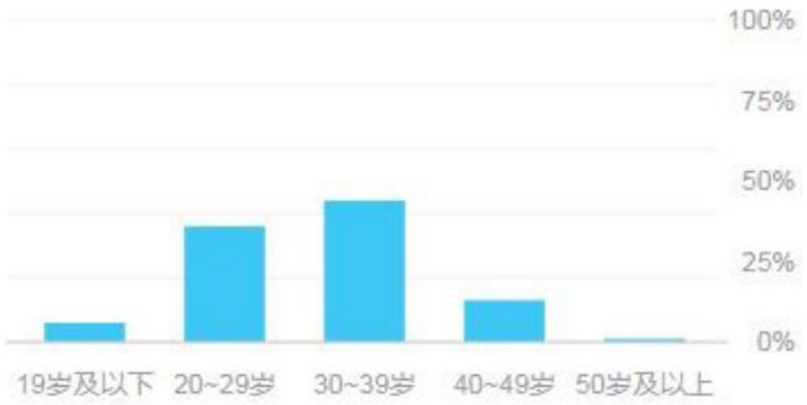
用户研究

数据挖掘用户特征

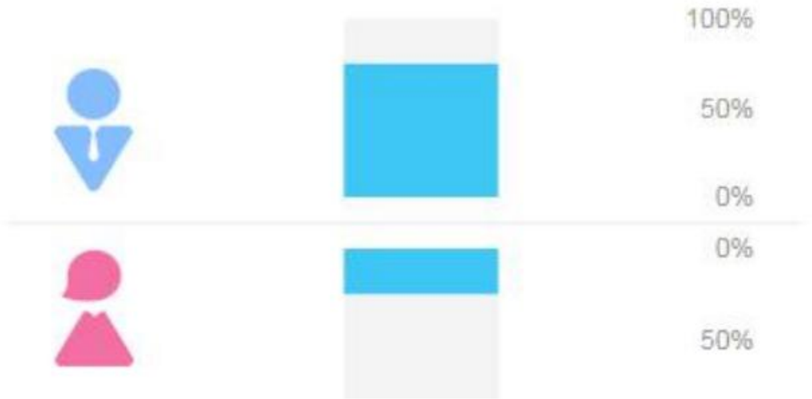
平台的用户画像

网易云课堂

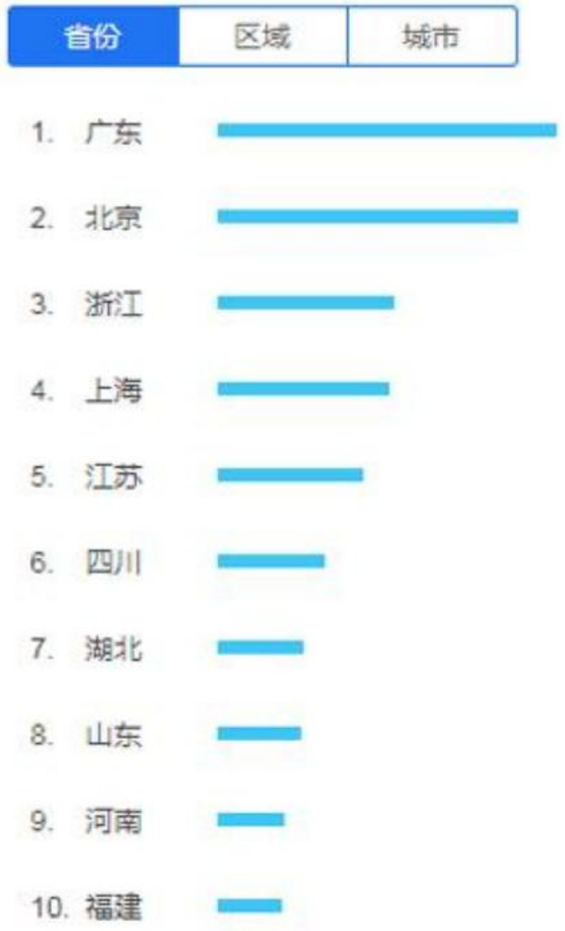
年龄分布：

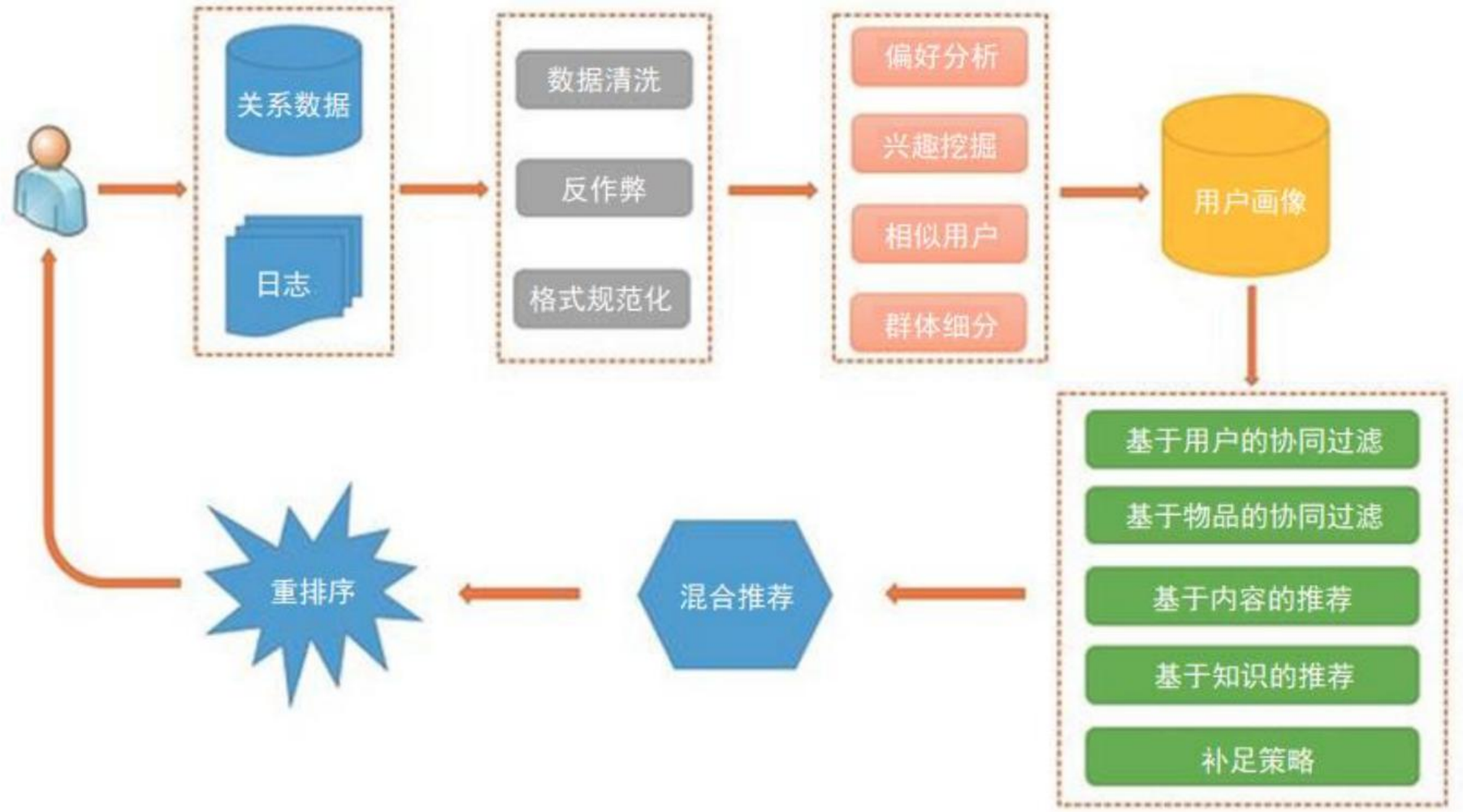


性别分布：

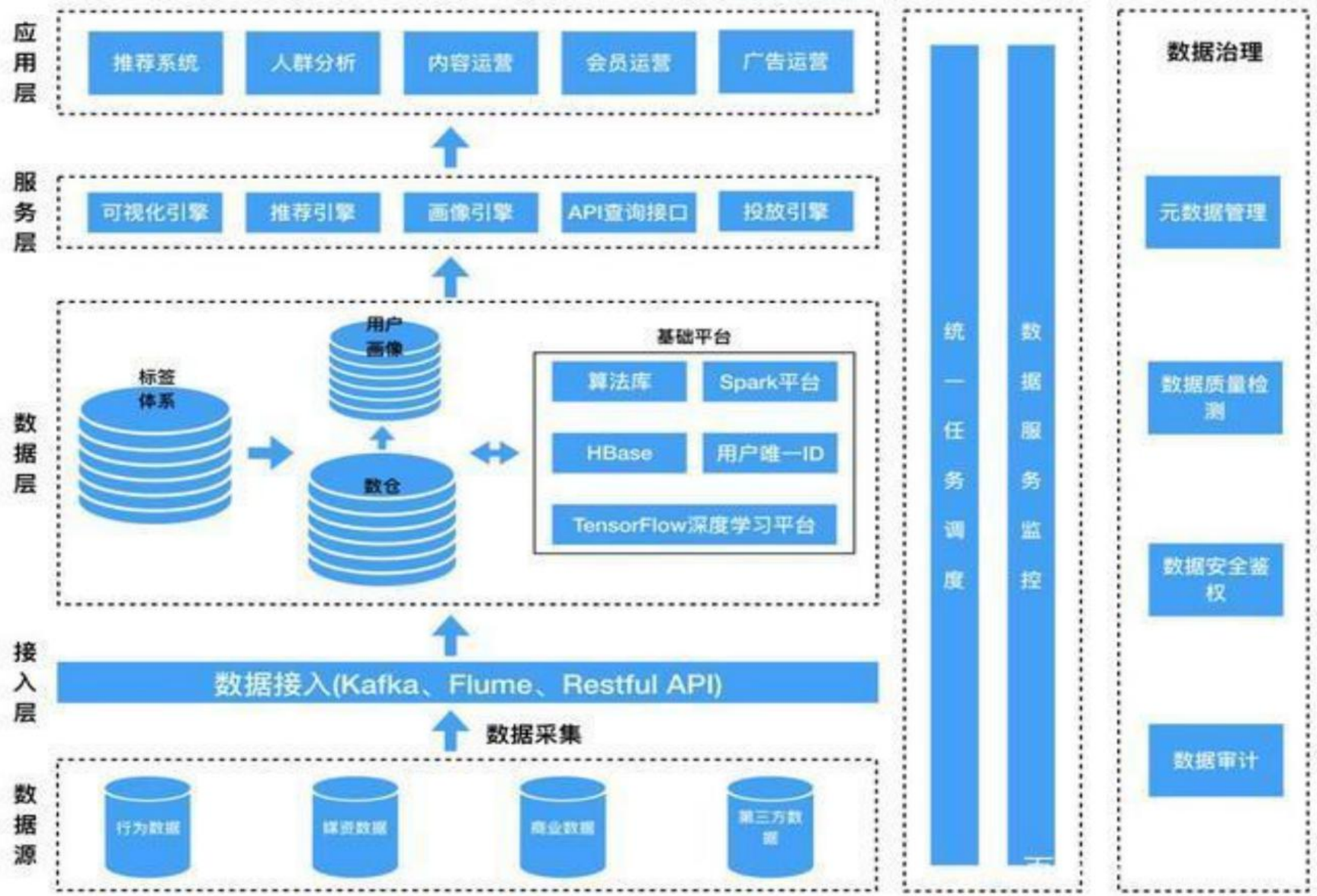


地域分布：




关系数据
日志
数据清洗
反作弊
格式规范化
偏好分析
兴趣挖掘
相似用户
群体细分
用户画像
基于用户的协同过滤
基于物品的协同过滤
基于内容的推荐
基于知识的推荐
补足策略
混合推荐
重排序

05 用户画像系统技术架构



58同城

转转

赶集网

安居客

其它

设备标签

1 Android D0001001
2 IOS D0001002
3 Winphone D0001003
4 其他D0001004

设备联网方式标签

WIFI D0002001
4G D0002002
3G D0002003
2G D0002004

搜索关键词标签

“麻将机|与神对话|家”

地域标签

省标签格式：ZPxxx->1
地市标签格式：ZCxxx->1

日志格式

, ^^05, 01, , , , 13aaa0a` /] 14, , - 5_] , 2(, (, (, (- , , , , , . (未知(. 2? 3> 5? 4/ @> 0> 2- 53? A> 4, @1/ >/ B1@=(- (- (, (, (. , , .) - ,)
, - , 26- 56- 3(- / 5*. . 3*- 2- *- - 1(_ki *] l l pmaadkp*dkmoa(赶集
网(=M' GEMa> dadt b2t b54> BBj h ? R, , l (=- , ! A11B! > ?! A2! =K! >! (- (0*- *- ((32, (54, (((上海市(上海市(0(/ (S ee(, (, (. (插屏(- (. (2(
未
知(- (, (, (, (, (, (, (, ((((((((((((, (111(. 0, (. 5, ((((((((((((=M' GEMa> dat b2t 544BBj h ? RKKl ((- (, (, (, (, (, (((i i [. 22/ . / 1/ [4, 2434, [. 3/ . 2115(
. , . ,) - ,), - , 26- 56- 3((

用户ID

imei	androidid	mac	idfa	openudid
imeimd5	androididmd5	macmd5	idfamd5	openudidmd5
imeisha1	androididsha1	macsha1	idfasha1	openudidsha1

用户ID获取

```
def getNotEmptyID(log: Logs): Option[String] = {
  log match {
    case v if v.imei.nonEmpty => Some("IMEI:" + Utils.formatIMEID(v.imei))
    case v if v.imeimd5.nonEmpty => Some("IMEIMD5:" + v.imeimd5.toUpperCase)
    case v if v.imeisha1.nonEmpty => Some("IMEISHA1:" + v.imeisha1.toUpperCase)

    case v if v.androidid.nonEmpty => Some("ANDROIDID:" + v.androidid.toUpperCase)
    case v if v.androididmd5.nonEmpty => Some("ANDROIDIDMD5:" + v.androididmd5.toUpperCase)
    case v if v.androididsha1.nonEmpty => Some("ANDROIDIDSHA1:" + v.androididsha1.toUpperCase)

    case v if v.mac.nonEmpty => Some("MAC:" + v.mac.replaceAll(":|-", "").toUpperCase)
    case v if v.macmd5.nonEmpty => Some("MACMD5:" + v.macmd5.toUpperCase)
    case v if v.macsha1.nonEmpty => Some("MACSHA1:" + v.macsha1.toUpperCase)

    case v if v.idfa.nonEmpty => Some("IDFA:" + v.idfa.replaceAll(":|-", "").toUpperCase)
    case v if v.idfamd5.nonEmpty => Some("IDFAMD5:" + v.idfamd5.toUpperCase)
    case v if v.idfasha1.nonEmpty => Some("IDFASHA1:" + v.idfasha1.toUpperCase)

    case v if v.openudid.nonEmpty => Some("OPENUDID:" + v.openudid.toUpperCase)
    case v if v.openudidmd5.nonEmpty => Some("OPENDUIDMD5:" + v.openudidmd5.toUpperCase)
    case v if v.openudidsha1.nonEmpty => Some("OPENUDIDSHA1:" + v.openudidsha1.toUpperCase)

    case _ => None
  }

}
```

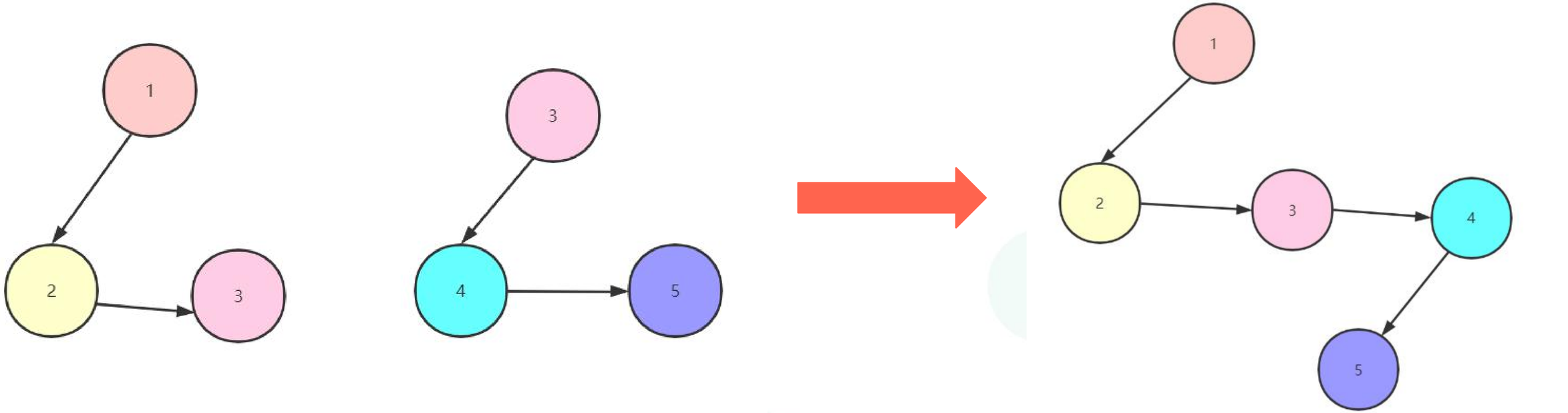
10 用户标签合并-ID Mapping

奈学教育

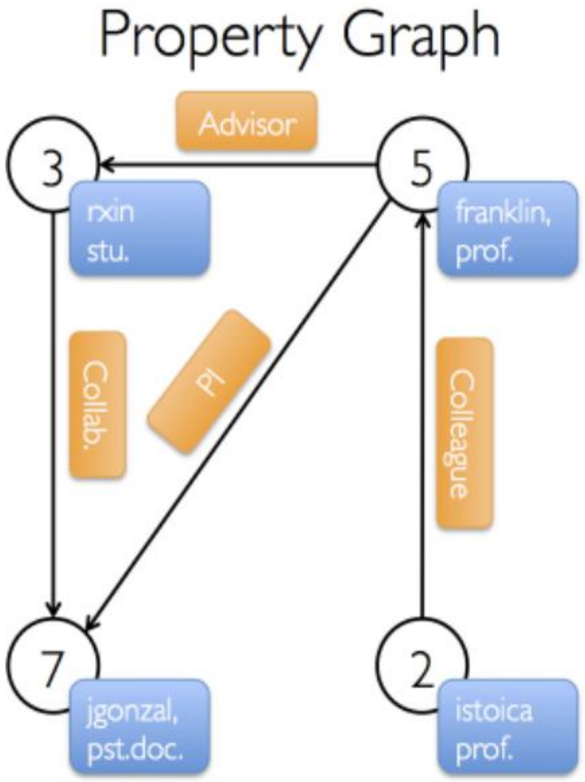
日志	imei	imeimd5	imeisha1	androidid	androididmd5	androididsha1	mac	macmd5	macsha1	idfa	idfamd5	idfasha1	openudid	openudidmd5	openudidsha1
1															
2															
3															
4															
5															
6															

ID Mapping：假设有一位用户张三，在第一个手机上使用百度地图， 在ipad上观看百度爱奇艺视频，在第二个手机上使用手机百度app, 在pc电脑上使用百度搜索，如何将同一个用户在这些不同端的用户信息聚合起来呢？ID-Mapping主要解决这个问题，用来关联ID信息。

02 分布式图计算实践


1
2
3
3
4
5
1
2
3
4
5

Spark GraphX是一个分布式图处理框架，它是基于Spark平台提供对图计算和图挖掘简洁易用的而丰富的接口，极大的方便了对分布式图处理的需求。



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

图的组成: G（P）=(V,E,P)

Vertex

Vertices: 

```
val users: RDD[(Long, (String, String))]
```

Edge

Edges: 

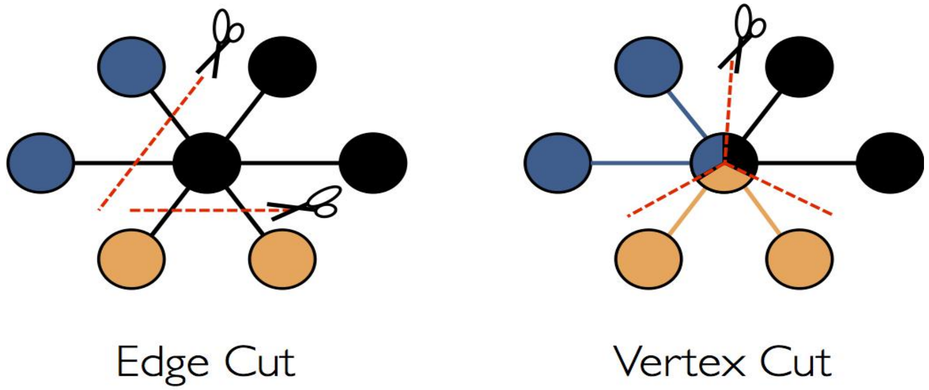
```
case class Edge[@specialized(Char, Int, Boolean, Byte, Long, Float, Double) ED] (
    var srcId: VertexId = 0,
    var dstId: VertexId = 0,
    var attr: ED = null.asInstanceOf[ED])
  extends Serializable {

  /**
```

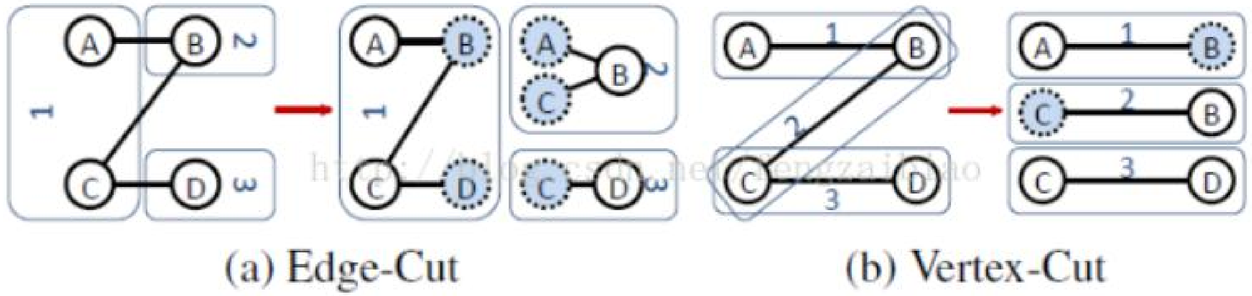
Property

```scala
// Assume the SparkContext has already been constructed
val sc: SparkContext
// Create an RDD for the vertices
val users: RDD[(VertexId, (String, String))] =
  sc.parallelize(Seq((3L, ("rxin", "student")), (7L, ("jgonzal", "postdoc")),
                     (5L, ("franklin", "prof")), (2L, ("istoica", "prof"))))
// Create an RDD for edges
val relationships: RDD[Edge[String]] =
  sc.parallelize(Seq(Edge(3L, 7L, "collab"),    Edge(5L, 3L, "advisor"),
                     Edge(2L, 5L, "colleague"), Edge(5L, 7L, "pi")))
// Define a default user in case there are relationship with missing user
val defaultUser = ("John Doe", "Missing")
// Build the initial Graph
val graph = Graph(users, relationships, defaultUser)
```

```scala
val graph: Graph[(String, String), String] // Constructed from above
// Count all users which are postdocs
graph.vertices.filter { case (id, (name, pos)) => pos == "postdoc" }.count
// Count all the edges where src > dst
graph.edges.filter(e => e.srcId > e.dstId).count
```

边分割和点分割





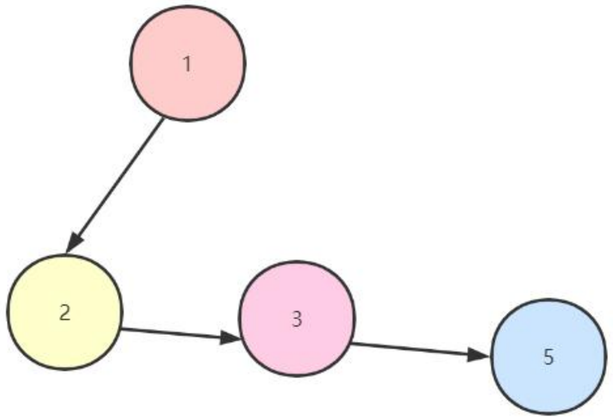
巨型图的存储总体上有边分割和点分割两种存储方式。2013 年，GraphLab2.0 将其存储方式由边分割变为点分割，在性能上取得重大提升，目前基本上被业界广泛接受并使用。

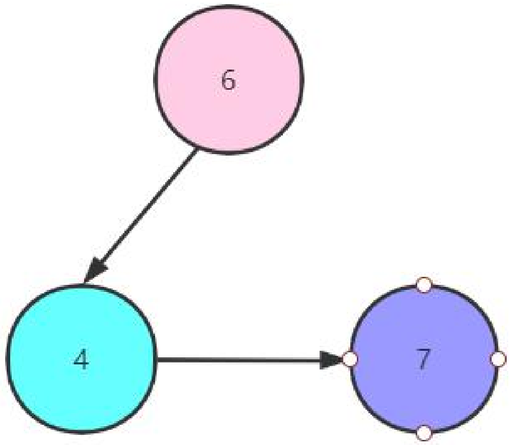
边分割（Edge-Cut）：每个顶点都存储一次，但有的边会被打断分到两台机器上。这样做的好处是节省存储空间；坏处是对图进行基于边的计算时，对于一条两个顶点被分到不同机器上的边来说，要跨机器通信传输数据，内网通信流量大。

点分割（Vertex-Cut）：每条边只存储一次，都只会出现在一台机器上。邻居多的点会被复制到多台机器上，增加了存储开销，同时会引发数据同步问题。好处是可以大幅减少内网通信量。

followers文件

1 2
2 3
3 5
4 6
7 6
6 7





Users

1,刘德华
2,张学友
3,黎明
4,郭富城
5,小李
6,小马
7,小张

郭富城,小马,小张

刘德华,黎明,小李,张学友

06 图计算实践之好友推荐小案例


```scala
//构建出来图有多种方式
val grapxh = GraphLoader.edgeListFile(sc, path = "followers.txt")
```

效果

```
(4,1)
(1,1)
(6,1)
(3,1)
(7,1)
(5,1)
(2,1)
```

```scala
val cc = grapxh.connectedComponents().vertices
```

效果

```
(4,4)
(6,4)
(7,4)

(1,1)
(3,1)
(5,1)
(2,1)
```

```scala
val users = sc.textFile( path = "users.txt").map(line => {
  val fields = line.split( regex = ",")
  (fields(0).toLong, fields(1))
})

//1,刘德华 join   1,1
//1 刘德华，1（代表的是同一个好友的那个id）
users.join(cc).map{
  case(id,(username,cclastid)) =>(cclastid,username)
}.reduceByKey( (x:String,y:String) => x + ","+ y)
  .foreach(tuple =>{
    println(tuple._2)
  })
```

效果

```
郭富城，小马，小张
刘德华，黎明，小李，张学友
```

03 分布式用户画像ID Mapping实践之标签合并

01 图计算之用户画像标签合并

步骤一：根据日志获取ID号：获取每一行日志里面的所有ID号(users)

数据格式：

Array(Tuple(1, Set("666","777")),
Tuple(2, Set("888","777")),Tuple(3, Set("777")))

步骤二：修改数据格式

数据格式：

666 1
777 1
888 2
777 2
777 3

步骤三：分组

数据格式：

666，{1}
777，{1，2，3}
888，{2}

步骤四：生成followers文件

数据格式：

1,1
1,1
1,2
1,3
2,2

步骤一和步骤四找好友

步骤五：用户关系表

数据格式：

9527　666,777,88

步骤六：获取之前标签并结果

数据格式：

666 (D00020005,2)　(ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2)
777 (D00020005,2)　(ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2)
888 (D00020005,2)　(ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2)

处理为：

666, List((D00020005,2)　(ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2))
777, List((D00020005,2)　(ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2))
888, List((D00020005,2)　(ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2))

步骤七：读取用户关系表

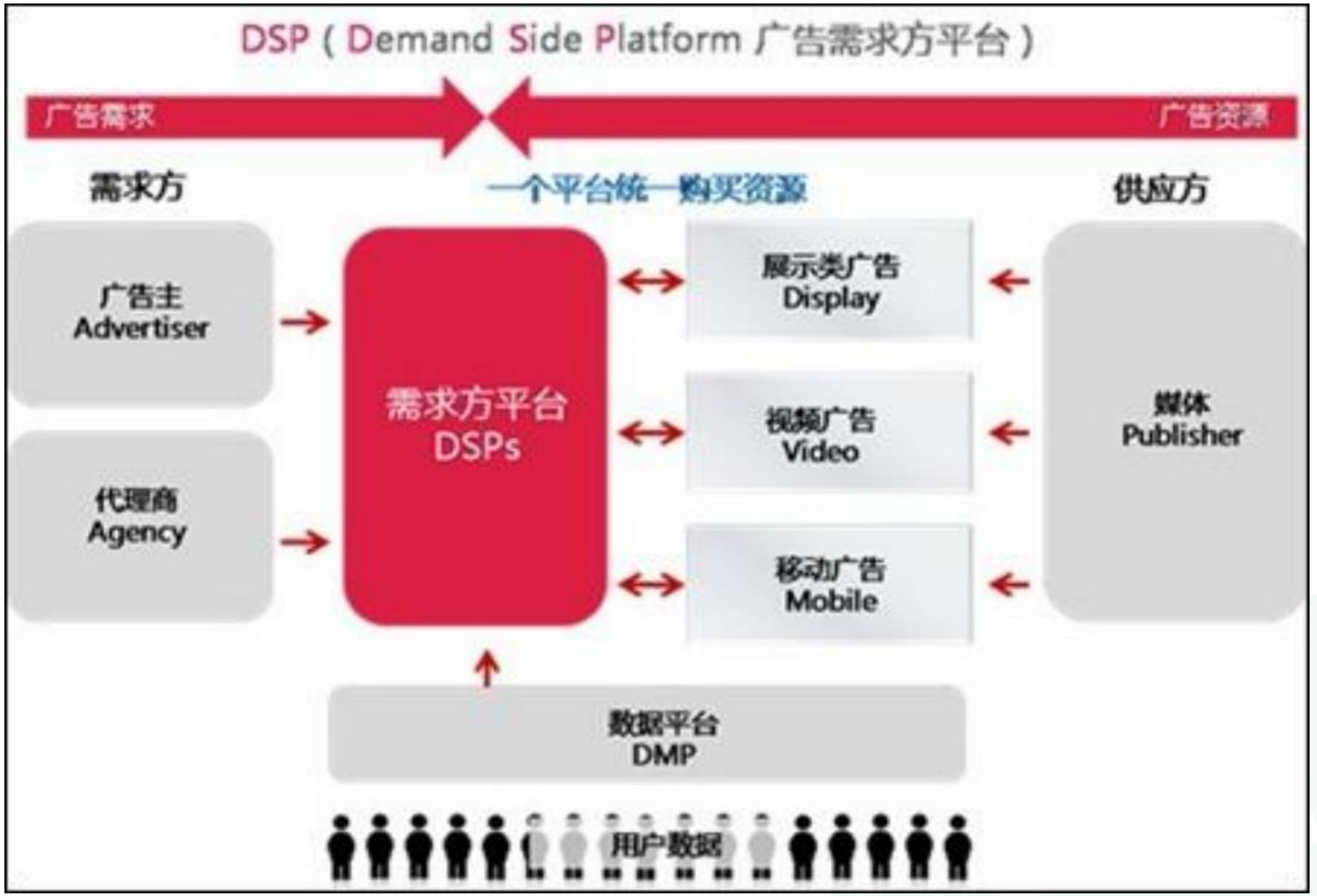
数据格式：

9527　666,777,88

步骤八：步骤六和步骤七join，然后再聚合

数据格式：

666	9527	join	666, List((D00020005,2) (ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2))
777	9527		777, List((D00020005,2) (ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2))
888	9527		888, List((D00020005,2) (ZC上海市,2) (APP赶集网,2)(D00010001,2)(ZP上海市,2))


DSP（Demand Side Platform 广告需求方平台）
广告需求
广告资源
需求方
一个平台统一购买资源
供应方
广告主
Advertiser
代理商
Agency
需求方平台
DSPs
展示类广告
Display
视频广告
Video
移动广告
Mobile
媒体
Publisher
数据平台
DMP
用户数据

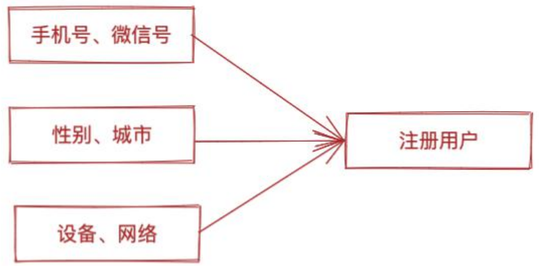
04 ID Mapping应用实践之风控领域

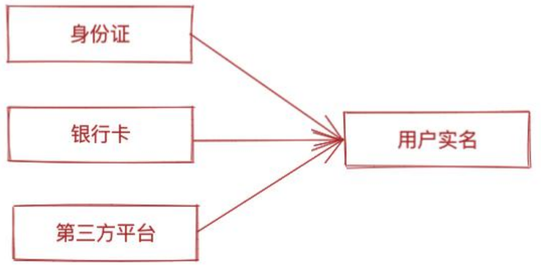
IDMapping

将不同维度的ID，查找关系映射到唯一ID的方法。最核心的用途之一是识别不同用户ID是同一个人。

电商互联网风控

- 用户端
  - 作弊用户
    - 突破平台的规则
    - 利用平台的漏洞
    - 马甲隐身
  - 正常
    - 行为正常
- 平台方
  - 跟踪用户ID
  - 识别用户ID行为
  - 打击用户ID





风控领域对ID的需求

随便玩玩

正常用户、具有一定价值

小打小闹

个体多马甲作战

团体、集群作战

团队集群作战

裂变繁殖

- 制造一些随便玩玩
- 打击个体
- 封堵团体

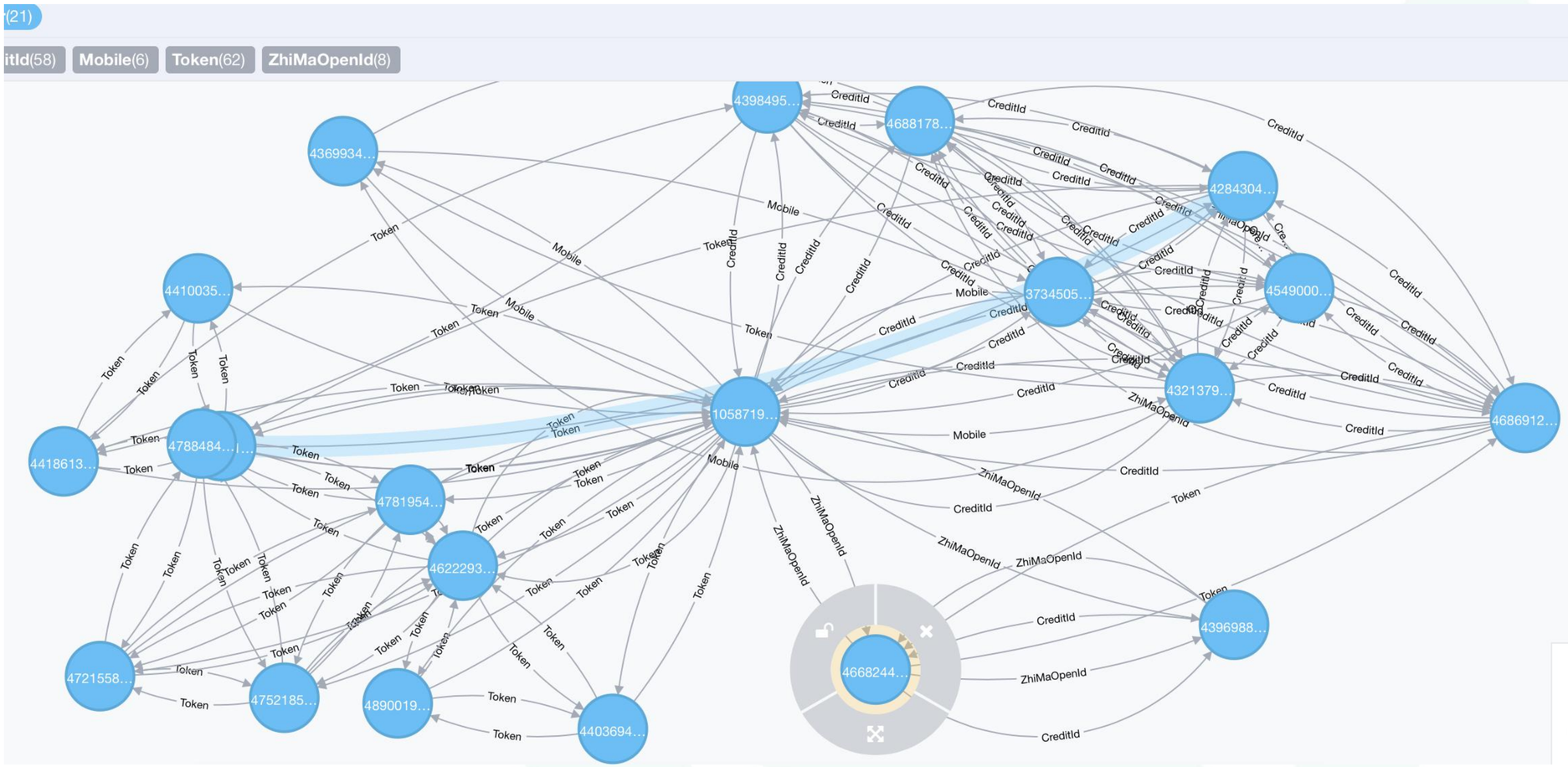
- 案例：拼多多一夜被薅疼

-月.，号凌晨一点至5点/，分，一则关于拼多多
-，，元无门槛优惠券的线报刷爆羊毛群。

事后：
平台对损失的大额变现的虚拟物品进行冻结并对变现实物的订单要求商户停止派发。

风控领域对ID的需求

- 拥有的数据
  - 用户注册信息
    - ✓ 设备信息
    - ✓ 手机信息
  - 行为过程信息
    - ✓ S dteE@信息
    - ✓ E 信息
  - 实名信息
    - ✓ 身份证信息
    - ✓ 微信Kj aE@
    - ✓ 支付宝Kj aE@



风控领域对ID的需求

- 拥有的数据
  - 用户注册信息
    - ✓ 设备信息
    - ✓ 手机信息
  - 行为过程信息
    - ✓ S deeE@信息
    - ✓ E 信息
  - 实名信息
    - ✓ 身份证信息
    - ✓ 微信Kj aE@
    - ✓ 支付宝Kj aE@

强关系

弱关系

用户ID如何利用？

- 强关系
  - 可以直接认为是同一个人
- 弱关系的使用？

IDMapping风控场景使用问题

- 基于OLTP业务规则
  - 当一个用户违规了，平台将他判定为黑名单
- 基于OLAP大数据挖掘
- 泛化挖掘

用户侧

违规

↓

账号被封

↓

换马甲

平台侧

识别违规

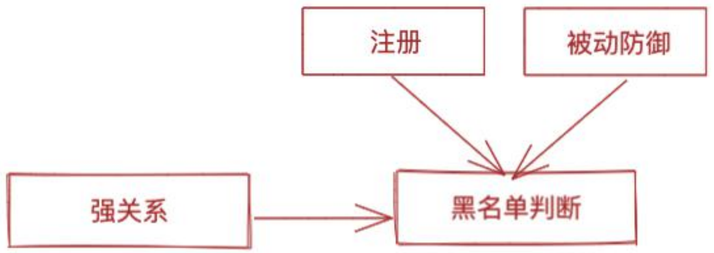
↓

黑名单体系

↓

识别马甲

误伤VS召回



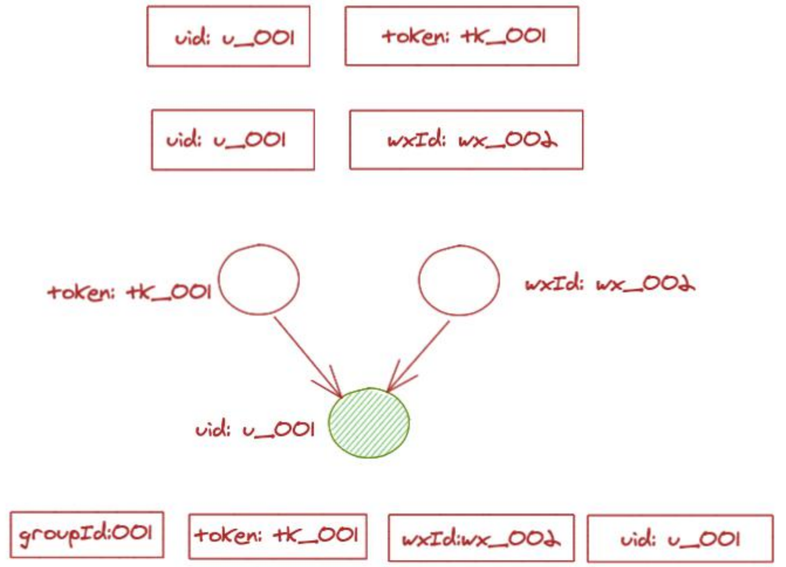
IDMapping风控场景使用问题

- 基于OLTP业务规则
- 基于OLAP大数据挖掘
  - 扩大数据范围
- 泛化挖掘

id数据没有交叉，所以产生两条单独数据，不是同一个人

数据DM层设计

- 窄表
  - id -> groupId
- 宽表
  - groupId -> idList



主动出击

- 使用groupId 进行连带处理
- 使用groupId 进行特征统计

IDMapping风控场景使用问题

- 基于OLTP业务规则
- 基于OLAP大数据挖掘
- 泛化挖掘

```
val uidCreditIdPair = spark.sql(
  s"""
     | select
     |   concat("uid-", uid) as uid,
     |   concat("cre-", credit_id) as creditCardId,
     |   murmurhash3(concat("uid-", uid)) as uidHash,
     |   murmurhash3(concat("cre-", credit_id)) as creditCardIdHash,
     |   count(*) as weight
     | from hdp_zhuanzhuan_rawdb_global.raw_user
     | where
     |   uid is not null
     |   and credit_id is not null
     |   and dt='$dateDash'
     | group by uid, credit_id
  """.stripMargin).persist
uidCreditIdPair.createOrReplaceTempView("uidCreditIdPair")
```

```
// 3. 构建图
val vertex = spark.sql(
  """
    | select
    |   vertexId,
    |   rawId
    | from (
    |   select
    |     uidHash as vertexId,
    |     uid as rawId
    |   from uidMobilePair
    |   union
    |   select
    |     mobileHash as vertexId,
    |     mobile as rawId
    |   from uidMobilePair
    |   union
    |   select
    |     uidHash as vertexId,
    |     uid as rawId
    |   from uidSpamTokenPair
    |   union
    |   select
    |     tokenHash as vertexId,
    |     token as rawId
    |   from uidSpamTokenPair
    |   union
    |   select
    |     creditCardIdHash as vertexId,
    |     creditCardId as rawId
    |   from uidCreditIdPair
    |   union
    |   select
    |     uidHash as vertexId,
    |     uid as rawId
    |   from uidCreditIdPair
    |   union
    |   select
    |     uidHash as vertexId,
    |     uid as rawId
    |   from uidIDCardPair
    |   union
    |   select
    |     idCardHash as vertexId,
    |     idcard as rawId
    |   from
    |     uidIDCardPair
    | ) t1
    | group by vertexId, rawId
  """.stripMargin)
vertex.createOrReplaceTempView("vertex")
```

```
val vertexDedup = spark.sql(
  """
    | select
    |   t1.vertexId,
    |   rawId
    | from
    |   vertex t1
    | join
    |   vertexNonCollisions t2
    | on t1.vertexId = t2.vertexId
  """.stripMargin)
vertexDedup.createOrReplaceTempView("vertexDedup")
```

```
val edgeDedup = spark.sql(
  """
    | select
    |   src,
    |   dest,
    |   sum(weight) as weight
    | from (
    |   select
    |     uidHash as src,
    |     mobileHash as dest,
    |     weight
    |   from uidMobilePair
    |   union
    |   select
    |     uidHash as src,
    |     tokenHash as dest,
    |     weight
    |   from uidSpamTokenPair
    |   union
    |   select
    |     uidHash as src,
    |     creditCardIdHash as dest,
    |     weight
    |   from uidCreditIdPair
    | ) t1
    | join vertexNonCollisions t2
    | on t1.src = t2.vertexId
    | join vertexNonCollisions t3
    | on t1.dest = t3.vertexId
    | group by src, dest
  """.stripMargin)
```

弱关系的处理

- 通过权重区别
  - 通过算法，作为特征输入
- 通过规则区分
  - 使用过多高频率才进入因子
  - 使用时间衰减函数

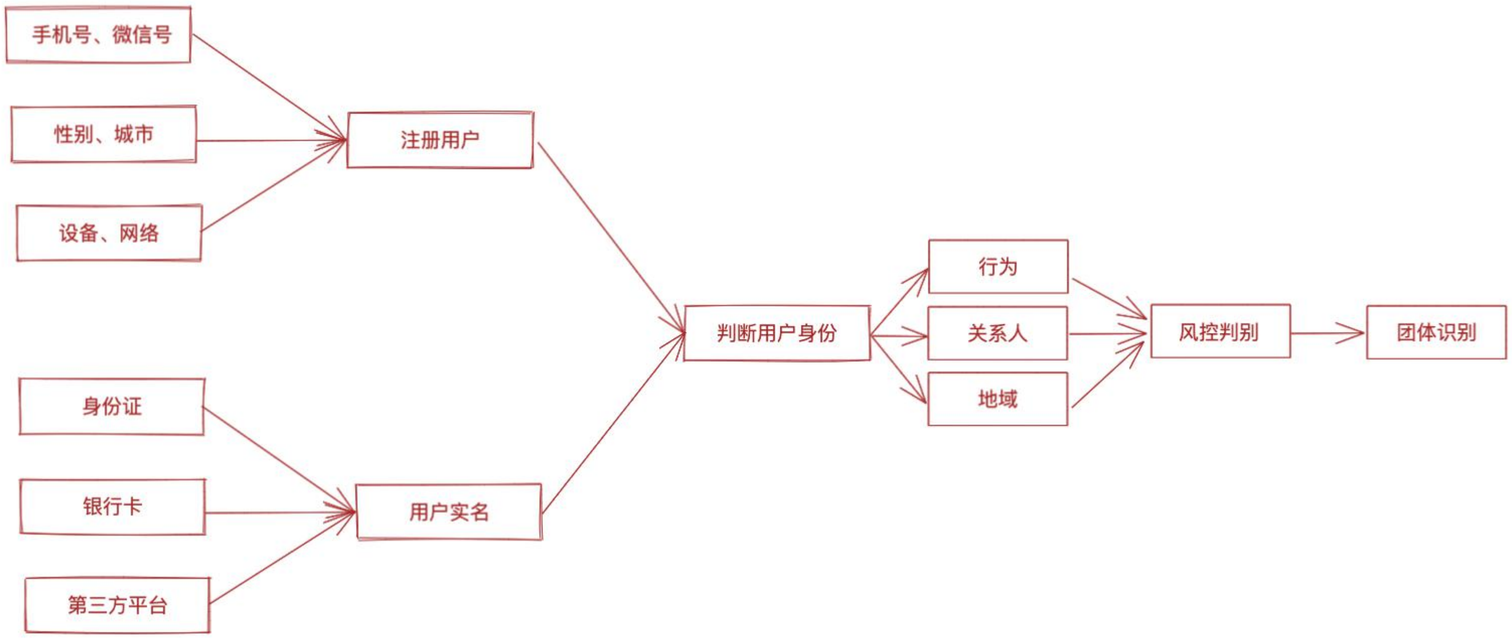
IDMapping风控场景使用问题

- 基于OLTP业务规则
- 基于OLAP大数据挖掘
- 泛化挖掘
  - 社群挖掘

图聚类算法

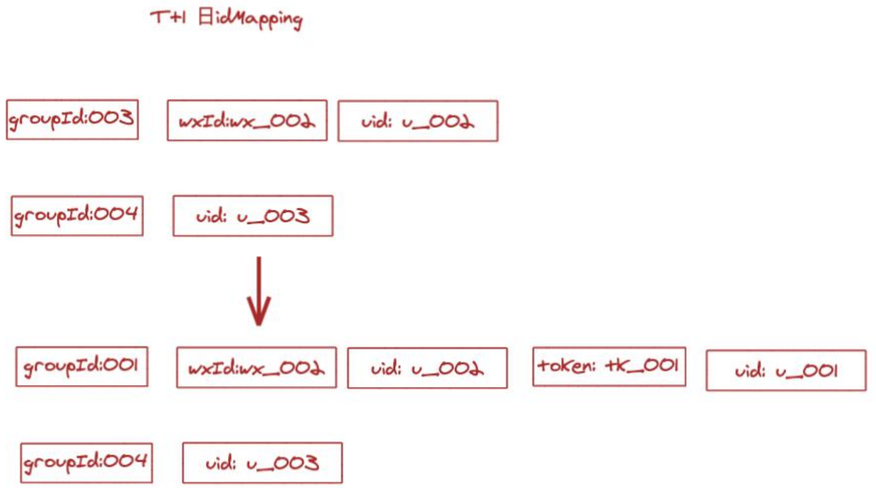
IDMapping风控场景使用问题

- 基于OLTP业务规则
- 基于OLAP大数据挖掘
- 泛化挖掘
  - 社群挖掘



IDMapping维护

- 每日全量计算
- 每日增量计算
  - 将新id-mapping跟旧的id-mapping进行合并操作



谢谢大家