AiCon
全球人工智能与机器学习技术大会

国美推荐引擎与算法持续部署实践

杨骥

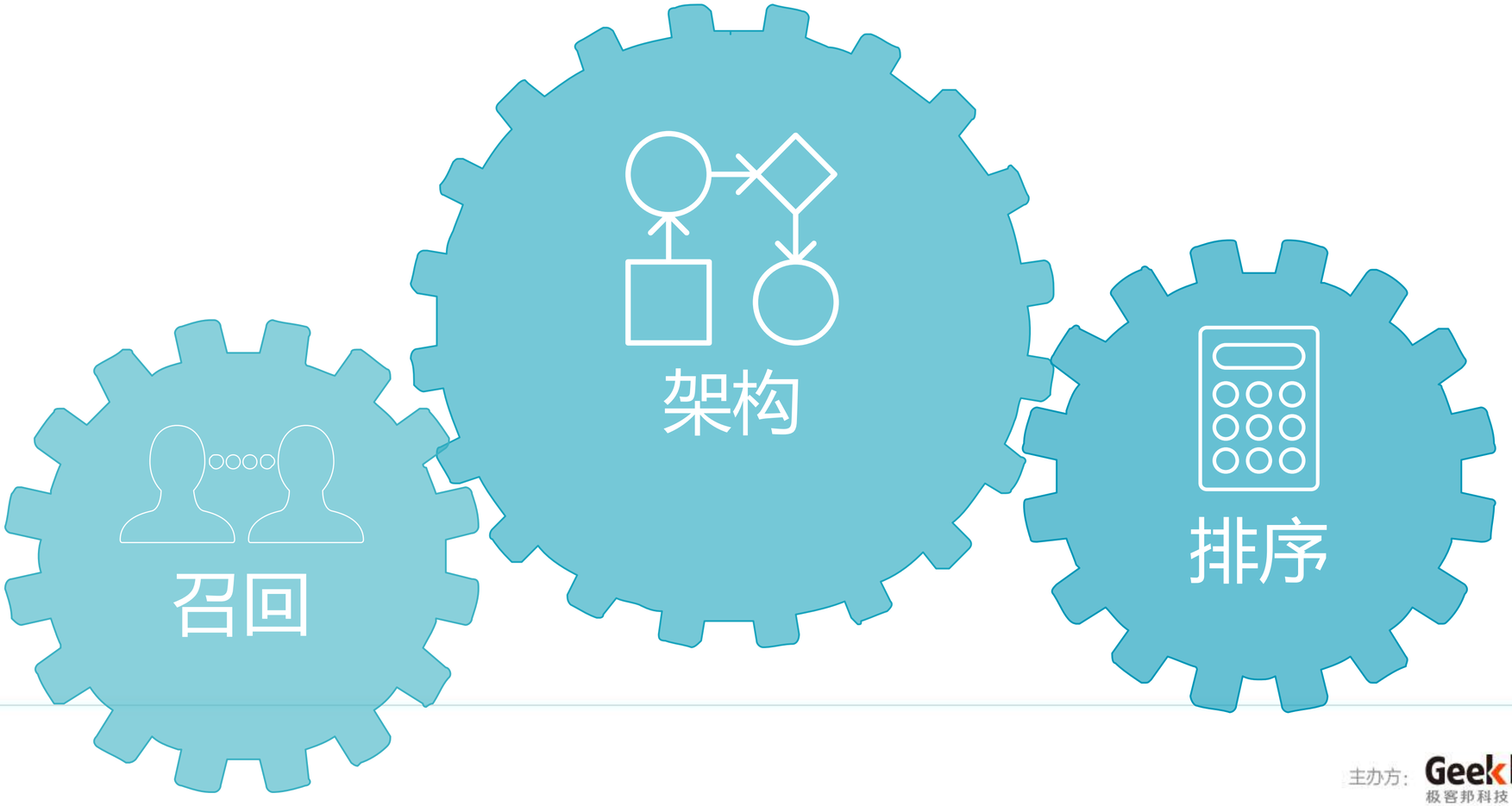
国美大数据中心副总监

主办方 Geekbang. 极客邦科技 InfoQ

TABLE OF CONTENTES

TABLE OF CONTENTES

推荐引擎



用户反馈

显示反馈

- 评分预测
- 出现在各大竞赛当中
- 样本确定
- 适用于Top-K推荐

商品

用户					
	1	?	?	2	?
	?	5	?	?	2
	3	?	?	1	?
	?	?	3	?	4
	4	?	?	?	5

真实值评分

隐示反馈

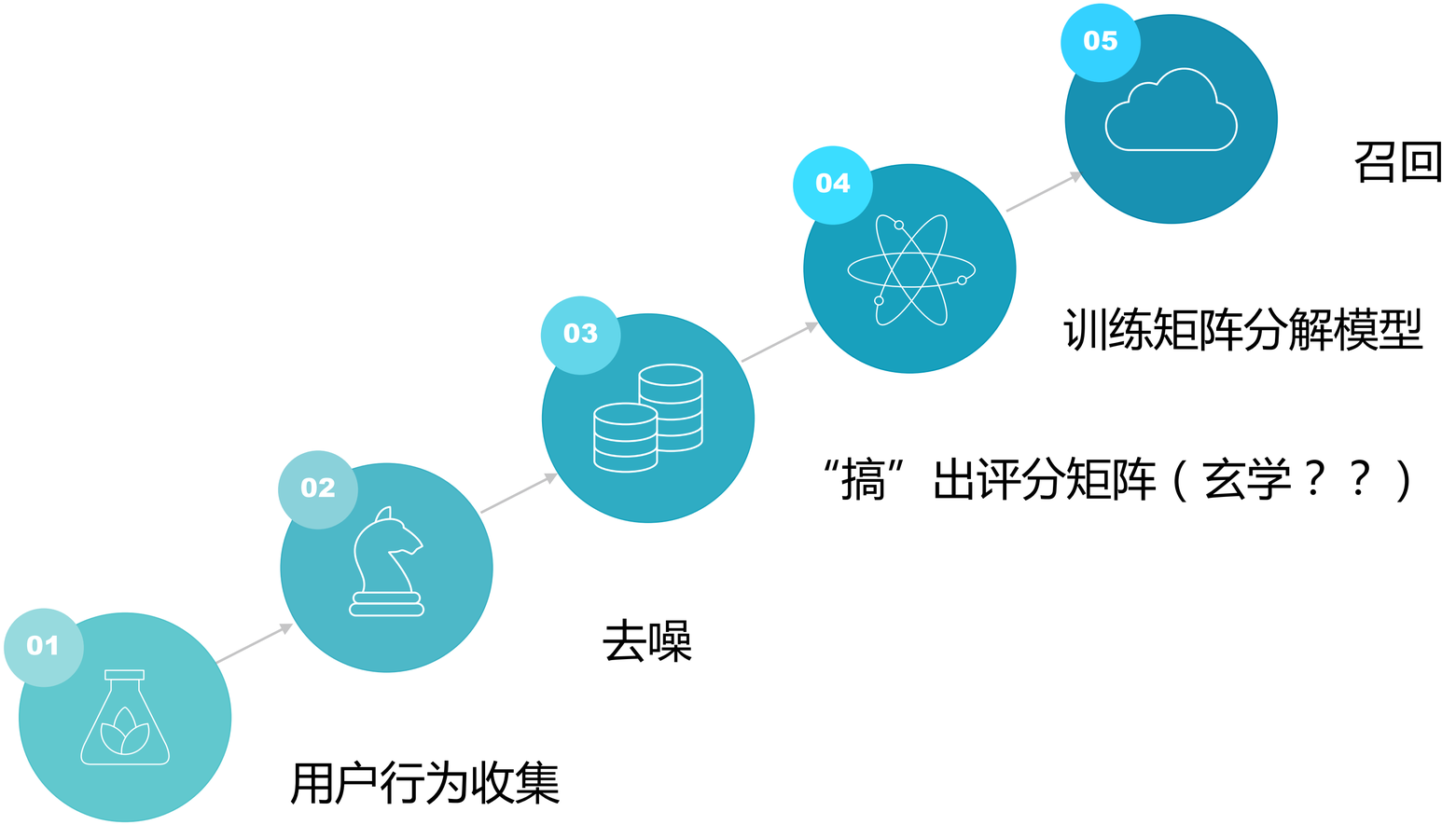
- 通常利用分类或者排序方式来进行建模
- 商品 → 用户
- 需要利用缺失的数据

商品

用户					
	1	0	0	1	0
	0	1	0	0	1
	1	0	0	1	0
	0	0	1	0	1
	1	0	0	0	1

0/1矩阵

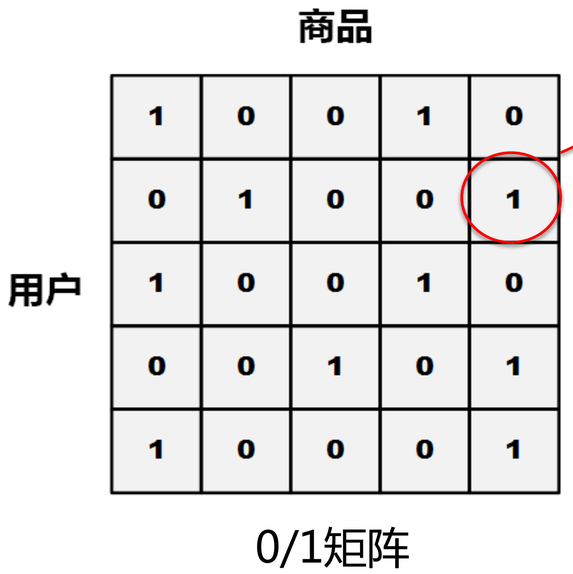
隐式反馈模型部署流程



Matrix Factorization模型

MF

本质上是一个线性隐向量模型



0/1矩阵

用户u和商品i产生交互

目标：训练得到每个用户u和商品i的隐向量

$$p_u \quad 1\times K$$

$$q_i$$

预测：根据内积得到用户u对商品i的感兴趣程度

$$\hat{y}_{ui} = \langle p_u, q_i \rangle$$

Matrix Factorization模型

MF

SVD++

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) + |R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}$$

Matrix Factorization模型

MF

将所有用户“跳”过的商品都看做负反馈

$$L(\Theta) = \sum_{(u,i)\in\mathcal{R}} (y_{ui} - \hat{y}_{ui})^2 + \omega_0 \sum_{(u,i)\notin\mathcal{R}} (0 - \hat{y}_{ui})^2$$

用户u点击过的商品

用户u“跳”过的商品

优点：对用户未点击过的商品进行了建模

不足：训练时数据量大，所有负反馈数据的权重都是相同的

Matrix Factorization模型

MF

考虑缺失数据样本（用户未点击）的权重

$$L(\Theta) = \sum_{u=1}^{M}\sum_{i=1}^{N} \omega_{ui}(y_{ui} - \hat{y}_{ui})^2 + \lambda\left(\sum_{u=1}^{M}\|p_u\|^2 + \sum_{i=1}^{N}\|q_i\|^2\right)$$

ω_{ui} 是每个样本权重，无论是用户点击还是跳过的

λ 是正则化项

y_{ui} 取值为1或者0

Matrix Factorization模型

MF

Alternating Least Square (ALS)

Step1

$$J_u = \|W^u(y_u - Qp_u)\|^2 + \lambda\|p_u\|^2$$

此时固定所有商品向量保持不变

$$\frac{\partial J_u}{\partial p_u} = 2Q^T W^u Q p_u - 2Q^T W^u y_u + 2\lambda p_u = 0$$

对p_u求导

$$p_u = (Q^T W^u Q + \lambda I)^{-1} Q^T W^u y_u$$

求解p_u的过程中有取逆运算

Matrix Factorization模型

MF

Alternating Least Square (ALS)

Step2

$$J_i = \left\| W^i(y_i - Pq_i) \right\|^2 + \lambda \|q_i\|^2$$

此时固定所有用户向量保持不变

$$\frac{\partial J_i}{\partial q_i} = 2P^T W^i P q_i - 2P^T W^i y_i + 2\lambda q_i = 0$$

对q_i求导

$$q_i = \left(P^T W^i P + \lambda I\right)^{-1} P^T W^i y_i$$

求解q_i的过程中有取逆运算

Matrix Factorization模型

MF

Generic Element-wise ALS

$$L(\Theta) = \sum_{u=1}^{M}\sum_{i=1}^{N} \omega_{ui}\left(y_{ui} - \hat{y}_{ui}^{f}\right)^2 + \lambda\left(\sum_{u=1}^{M}\|p_u\|^2 + \sum_{i=1}^{N}\|q_i\|^2\right)$$

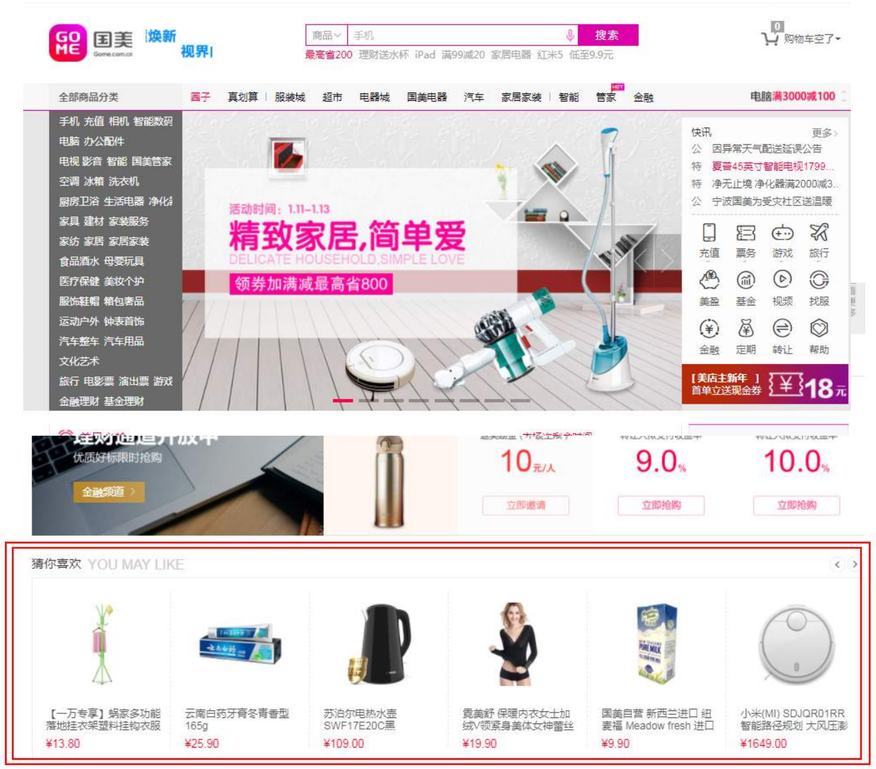
$$\Rightarrow \quad \frac{\partial L}{\partial p_{uf}} = -2\sum_{i=1}^{N}\left(y_{ui} - \hat{y}_{ui}^{f}\right)\omega_{ui}q_{if} + 2p_{uf}\sum_{i=1}^{N}\omega_{ui}{q_{if}}^2 + 2\lambda P_{uf}$$

$$\Rightarrow \quad p_{uf} = \frac{\sum_{i=1}^{N}\left(y_{ui} - \hat{y}_{ui}^{f}\right)\omega_{ui}q_{if}}{\sum_{i=1}^{N}\omega_{ui}{q_{if}}^2 + \lambda} \qquad \Rightarrow \quad q_{if} = \frac{\sum_{u=1}^{M}\left(y_{ui} - \hat{y}_{ui}^{f}\right)\omega_{ui}p_{uf}}{\sum_{u=1}^{M}\omega_{ui}{p_{uf}}^2 + \lambda}$$

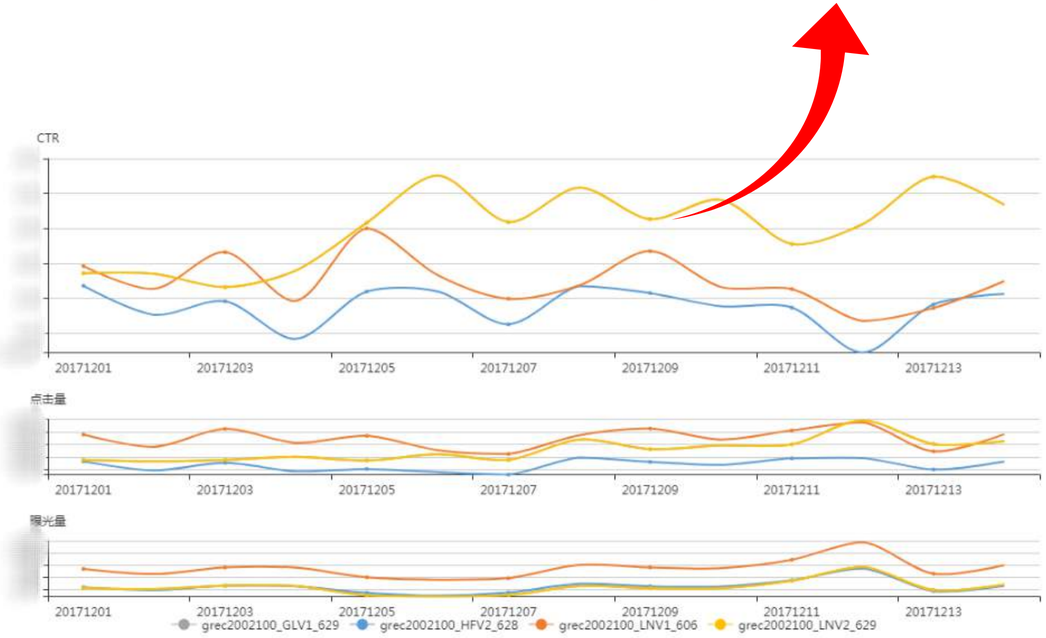
Matrix Factorization模型

MF

Generic Element-wise ALS



点击率提升：18.26%



Matrix Factorization模型

MF

Generic Element-wise ALS

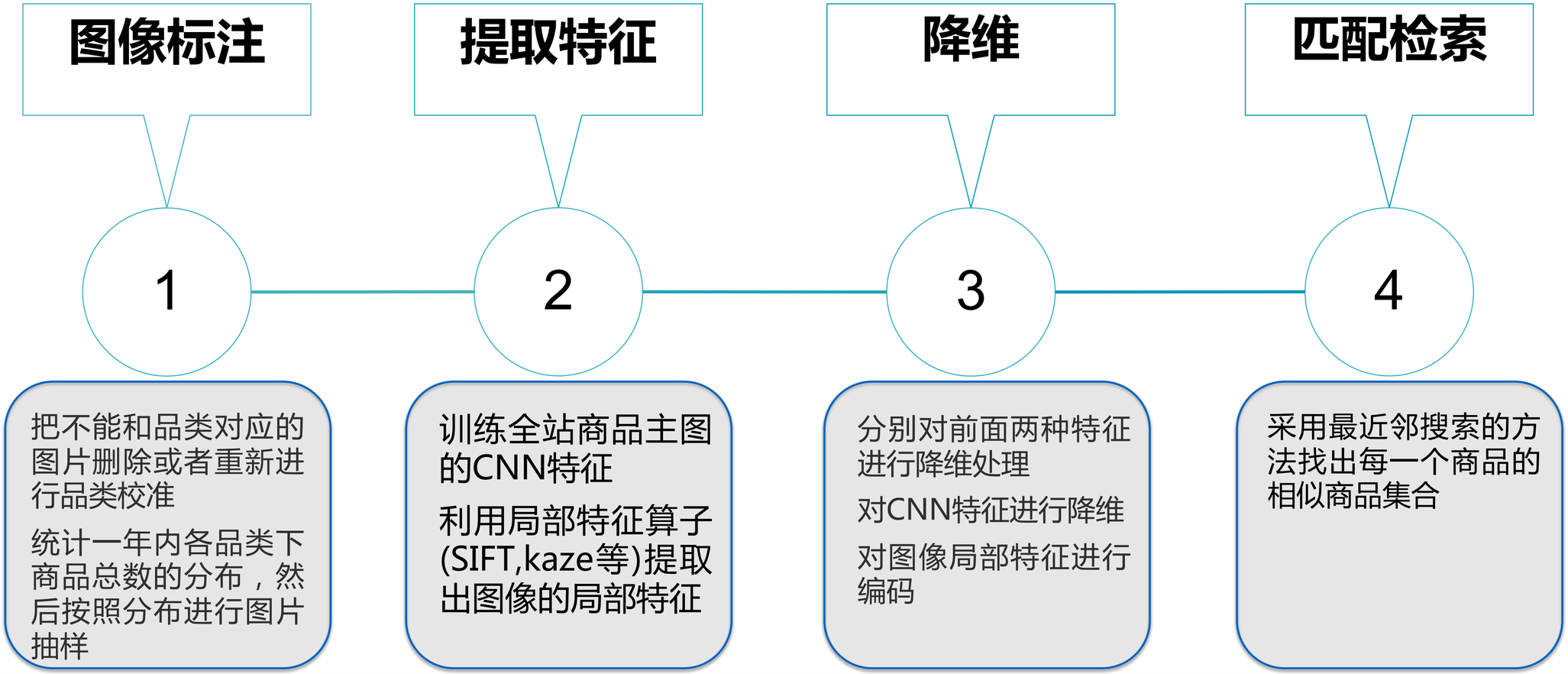
He X, Zhang H, Kan M Y, et al. Fast Matrix Factorization for Online Recommendation with Implicit Feedback[J]. 2017:549-558.

Rendle S, Gantner Z, Freudenthaler C, et al. Fast context-aware recommendations with factorization machines[C]// International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2011:635-644.

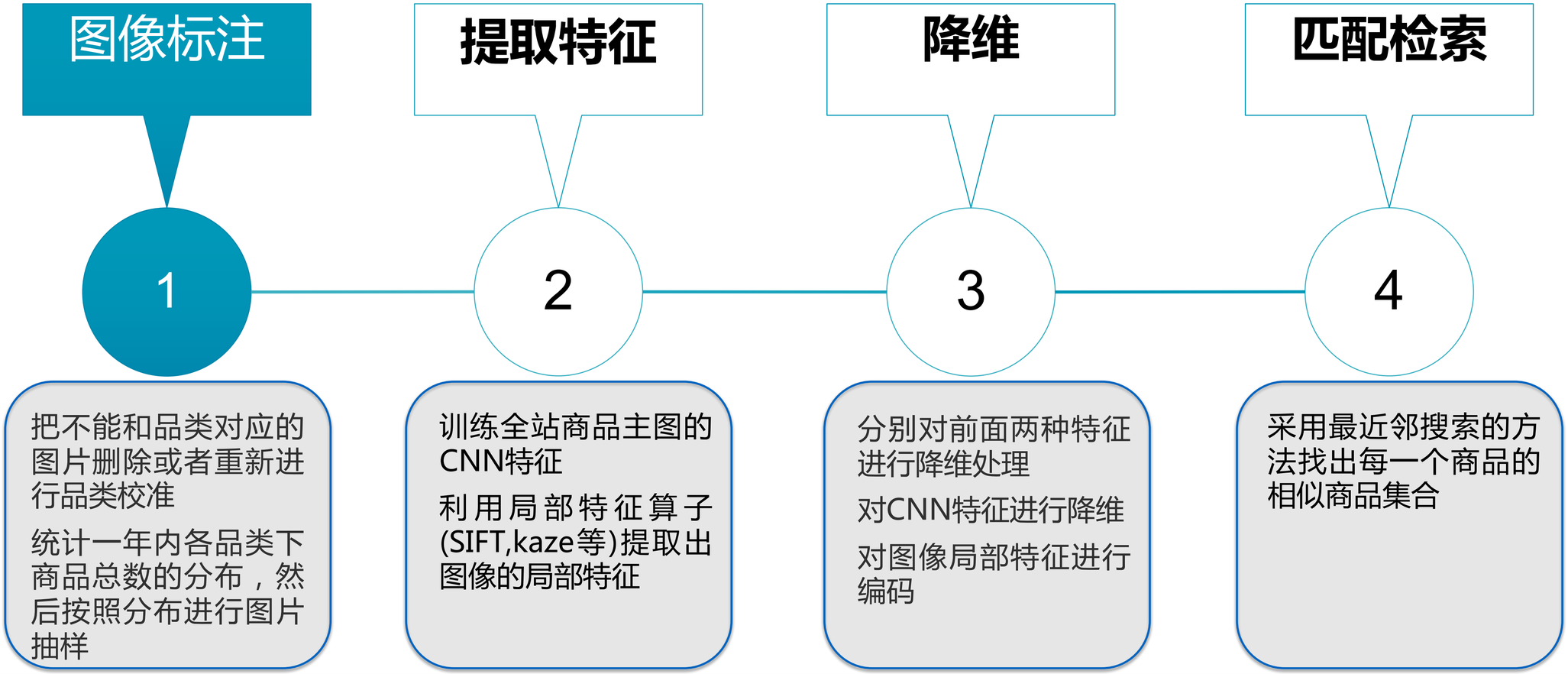
TABLE OF CONTENTES

TABLE OF CONTENTES

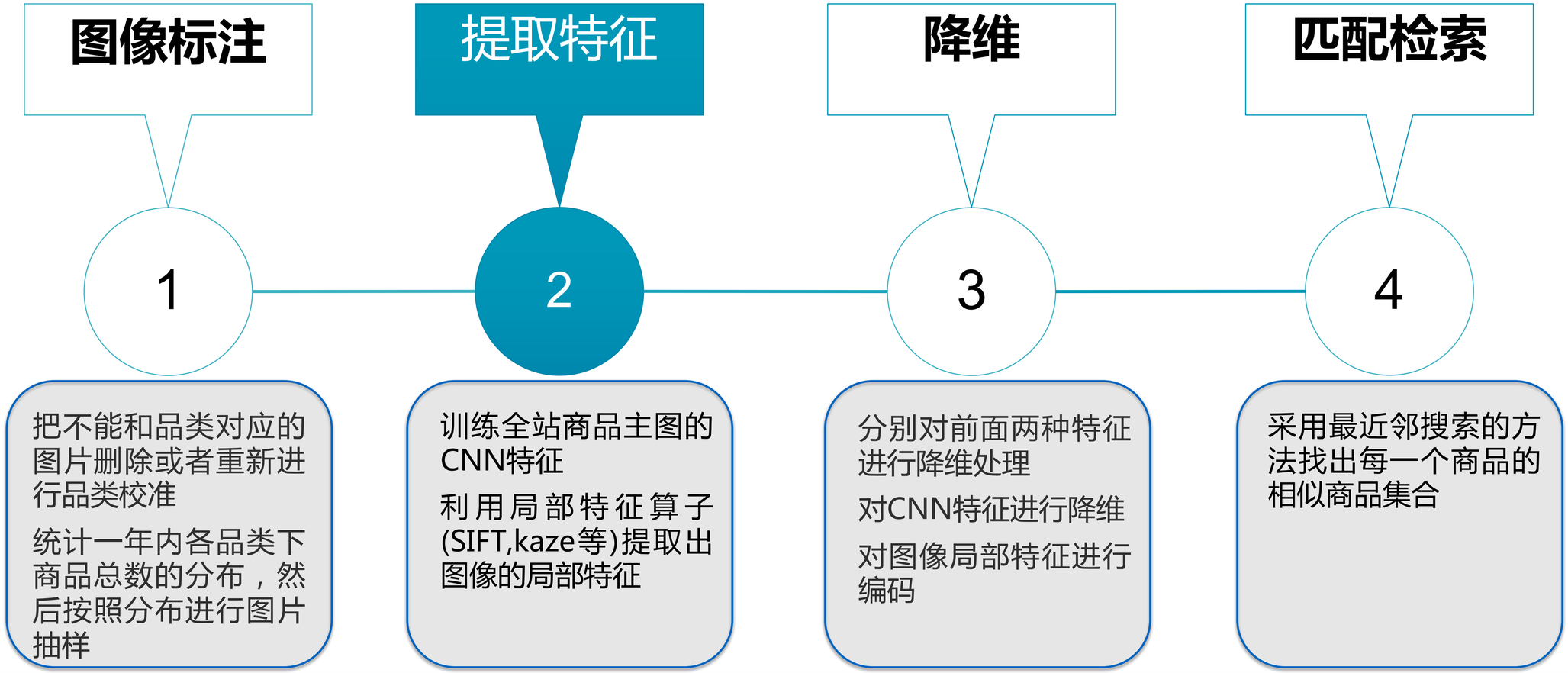
图像检索流程


图像标注
提取特征
降维
匹配检索
1
2
3
4
把不能和品类对应的图片删除或者重新进行品类校准
统计一年内各品类下商品总数的分布，然后按照分布进行图片抽样
训练全站商品主图的CNN特征
利用局部特征算子(SIFT,kaze等)提取出图像的局部特征
分别对前面两种特征进行降维处理
对CNN特征进行降维
对图像局部特征进行编码
采用最近邻搜索的方法找出每一个商品的相似商品集合

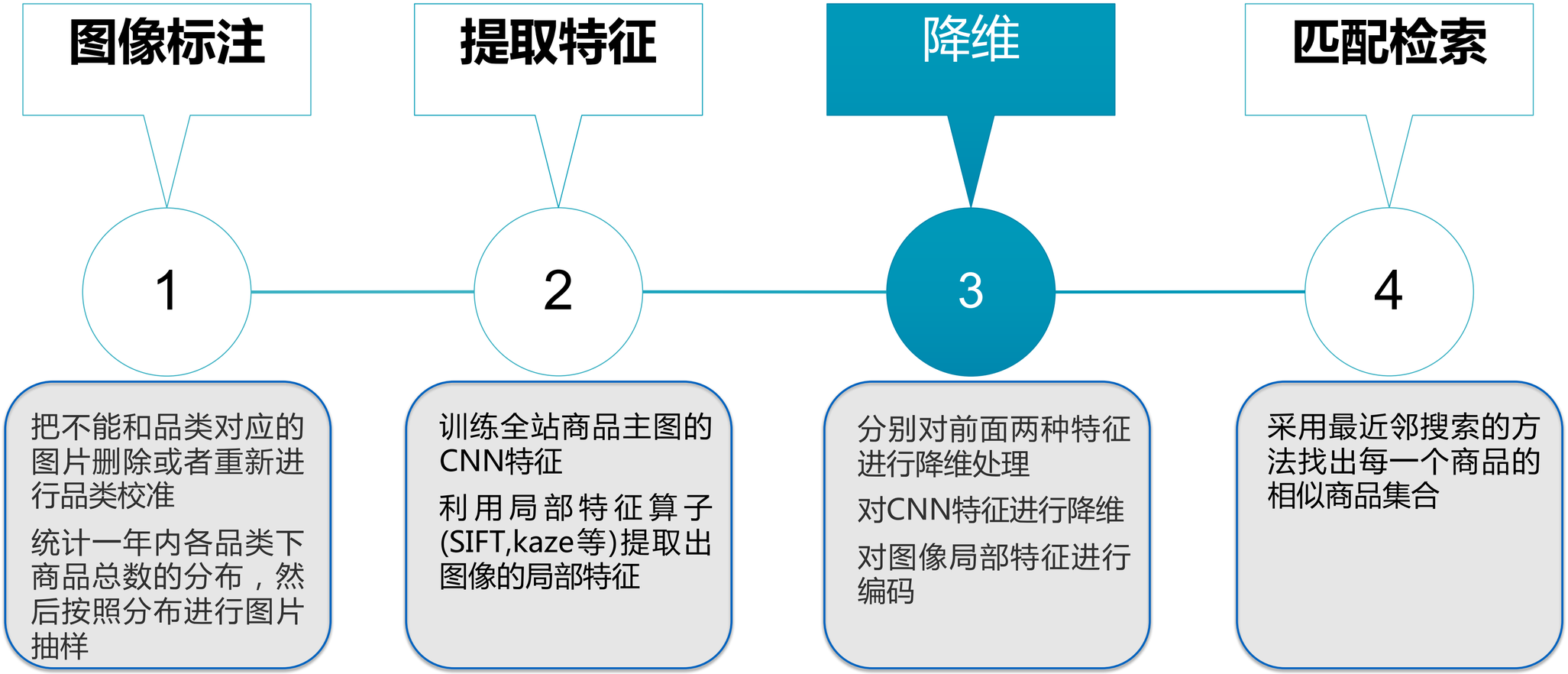
图像检索流程


图像标注
提取特征
降维
匹配检索
1
2
3
4
把不能和品类对应的图片删除或者重新进行品类校准
统计一年内各品类下商品总数的分布，然后按照分布进行图片抽样
训练全站商品主图的CNN特征
利用局部特征算子(SIFT,kaze等)提取出图像的局部特征
分别对前面两种特征进行降维处理
对CNN特征进行降维
对图像局部特征进行编码
采用最近邻搜索的方法找出每一个商品的相似商品集合

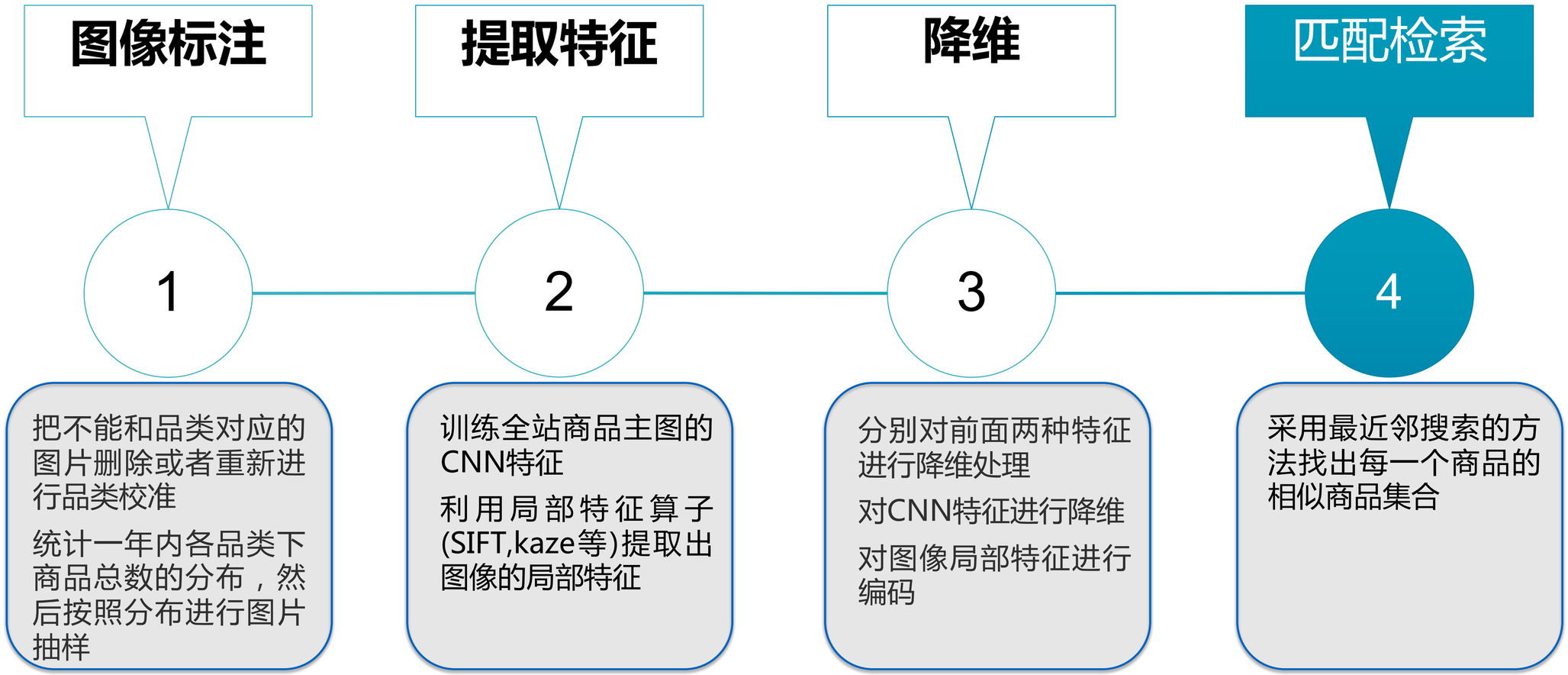
图像检索流程


图像标注
提取特征
降维
匹配检索
1
2
3
4
把不能和品类对应的图片删除或者重新进行品类校准
统计一年内各品类下商品总数的分布，然后按照分布进行图片抽样
训练全站商品主图的CNN特征
利用局部特征算子(SIFT,kaze等)提取出图像的局部特征
分别对前面两种特征进行降维处理
对CNN特征进行降维
对图像局部特征进行编码
采用最近邻搜索的方法找出每一个商品的相似商品集合

图像检索流程


图像标注
提取特征
降维
匹配检索
1
2
3
4
把不能和品类对应的图片删除或者重新进行品类校准
统计一年内各品类下商品总数的分布，然后按照分布进行图片抽样
训练全站商品主图的CNN特征
利用局部特征算子(SIFT,kaze等)提取出图像的局部特征
分别对前面两种特征进行降维处理
对CNN特征进行降维
对图像局部特征进行编码
采用最近邻搜索的方法找出每一个商品的相似商品集合

图像检索流程


图像标注
提取特征
降维
匹配检索
1
2
3
4
把不能和品类对应的图片删除或者重新进行品类校准
统计一年内各品类下商品总数的分布，然后按照分布进行图片抽样
训练全站商品主图的CNN特征
利用局部特征算子(SIFT,kaze等)提取出图像的局部特征
分别对前面两种特征进行降维处理
对CNN特征进行降维
对图像局部特征进行编码
采用最近邻搜索的方法找出每一个商品的相似商品集合

国美拍照推荐

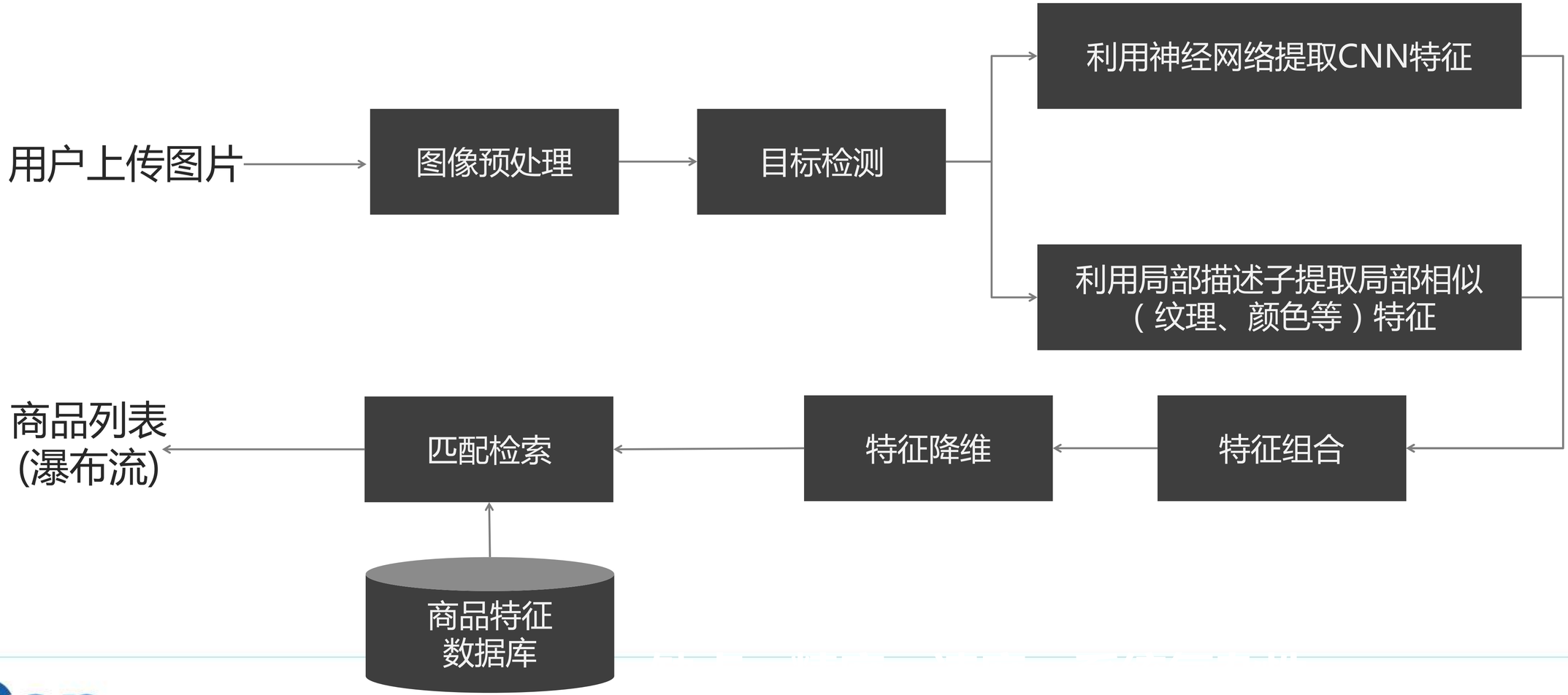
拍照购

满足消费者在各种场景下的即时购物需求

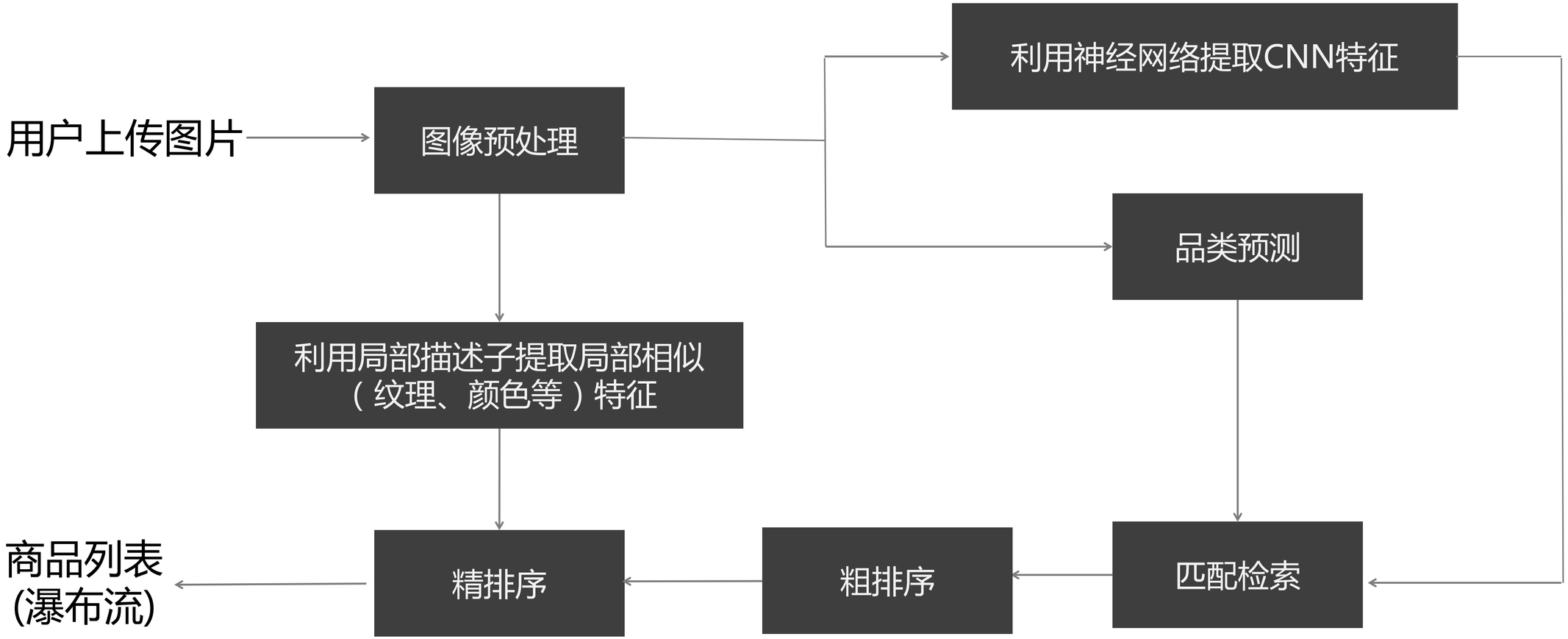
实现效果

根据用户上传图片，通过特征向量的计算和匹配，找到用户感兴趣的商品，满足消费者在各种场景下的即时购物需求

拍照推荐流程


用户上传图片
图像预处理
目标检测
利用神经网络提取CNN特征
利用局部描述子提取局部相似
（纹理、颜色等）特征
特征组合
特征降维
匹配检索
商品列表
(瀑布流)
商品特征
数据库

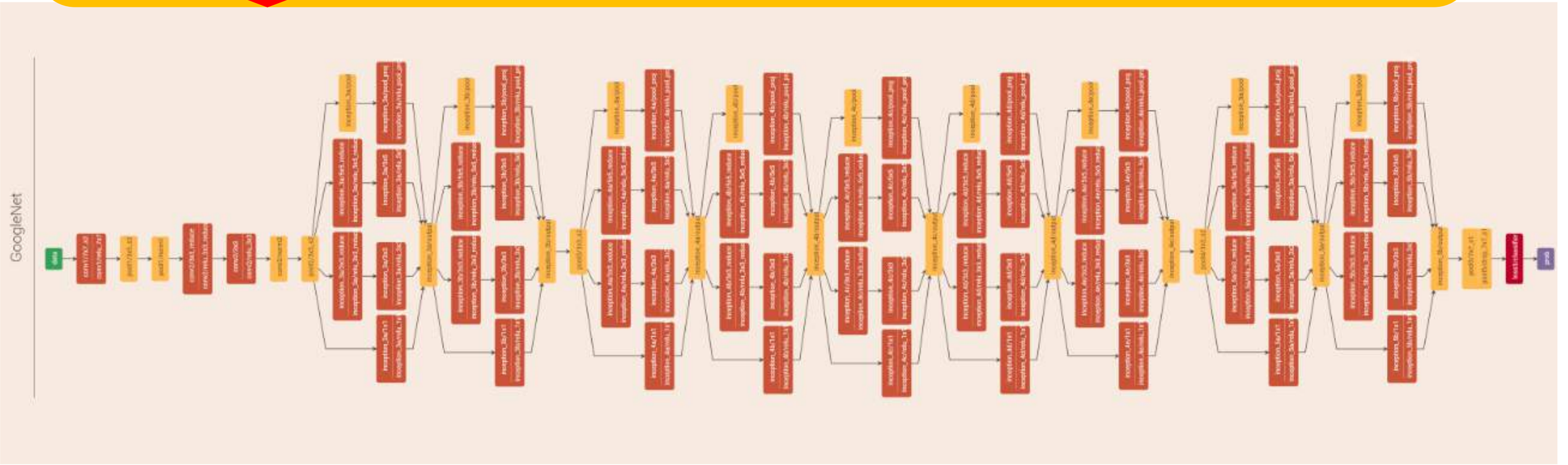
拍照推荐流程


用户上传图片
图像预处理
利用神经网络提取CNN特征
品类预测
利用局部描述子提取局部相似
（纹理、颜色等）特征
匹配检索
粗排序
精排序
商品列表
(瀑布流)

CNN特征抽取

CNN

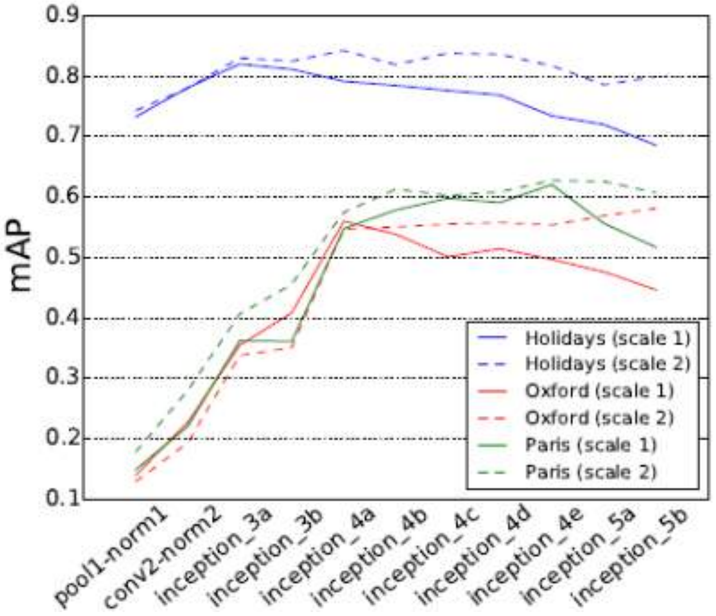
利用GoogleNet提取商品主图特征



CNN特征抽取

CNN

GoogleNet的不同层对召回结果的影响



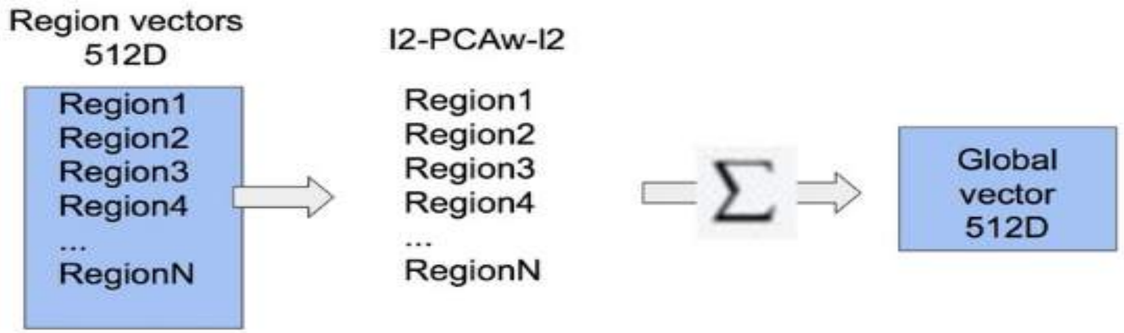
不能只用最高层的特征做检索

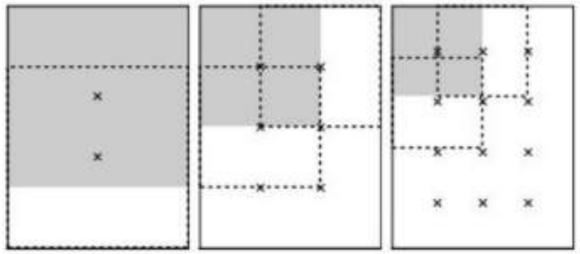
高层的语义特征丰富，但是缺少了细节信息

CNN特征抽取

CNN

特征融合

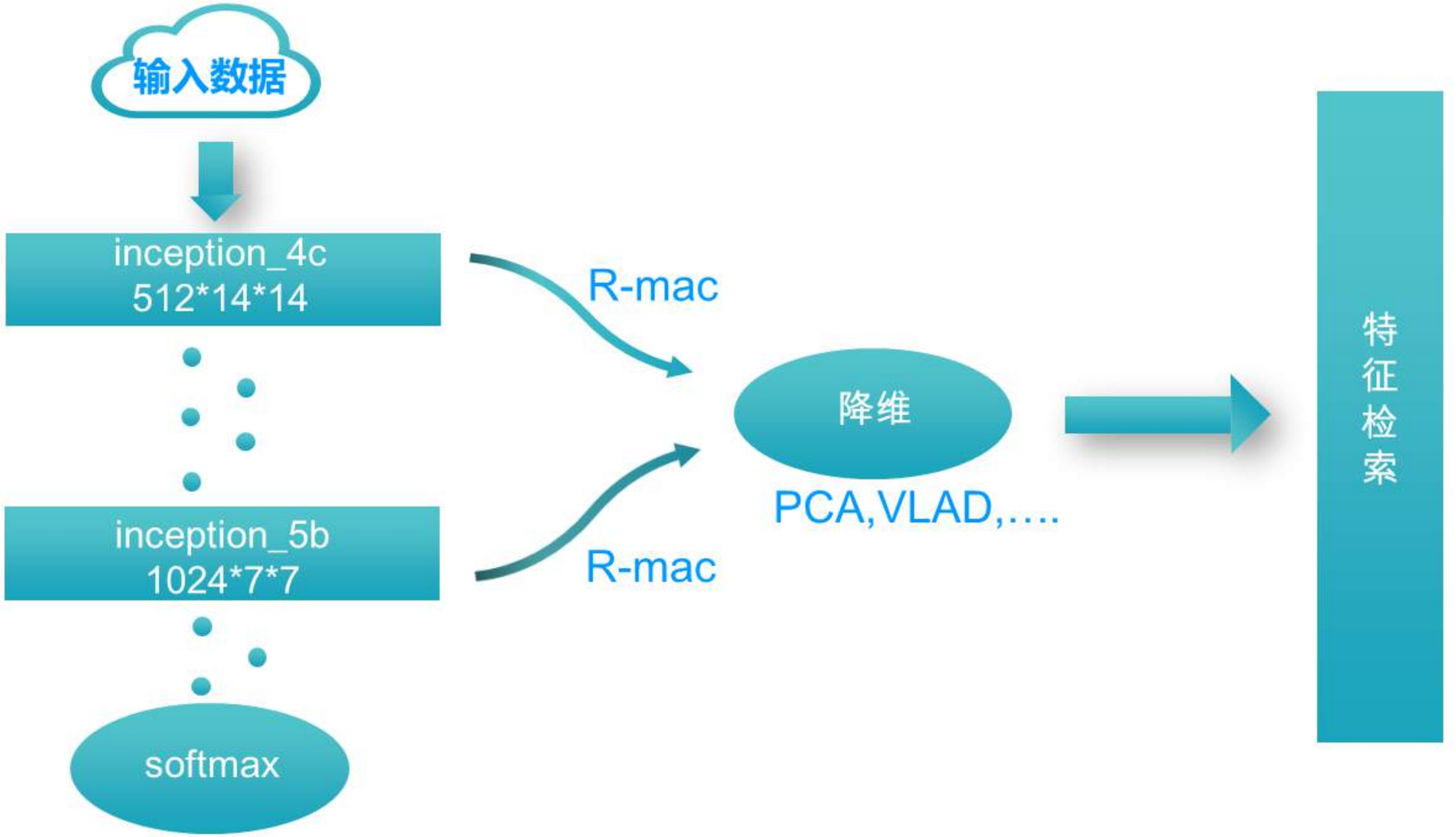




对特征图进行多尺度类似滑窗处理，然后进行相加

regions的宽=高=2*min(W,H)/(l+1)，共采样 l*(l+1)个regions, l分别为1，2，3

特征融合


输入数据
inception_4c
512*14*14
inception_5b
1024*7*7
softmax
R-mac
R-mac
降维
PCA,VLAD,....
特征检索

CNN特征抽取

R-MAC

Regional Maximum Activation of Convolutions

Li Y, Xu Y, Wang J, et al. MS-RMAC: Multi-Scale Regional Maximum Activation of Convolutions for Image Retrieval[J]. IEEE Signal Processing Letters, 2017, PP(99):1-1.

利用图像内容进行召回

点击率提升：18.26%

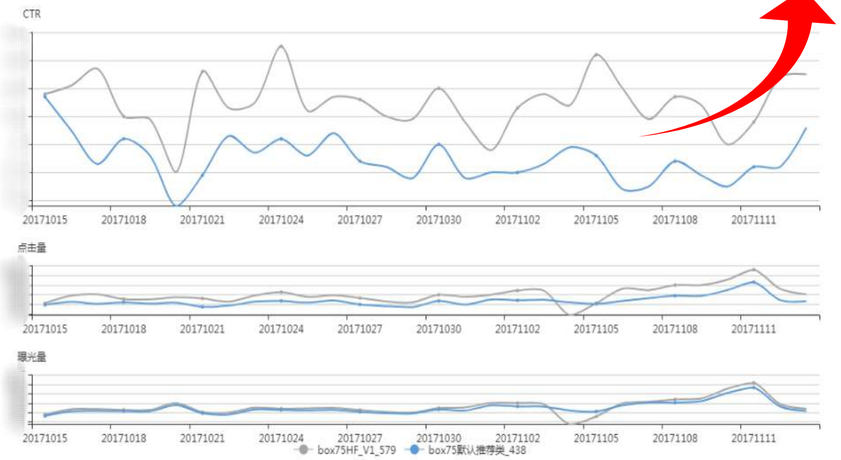


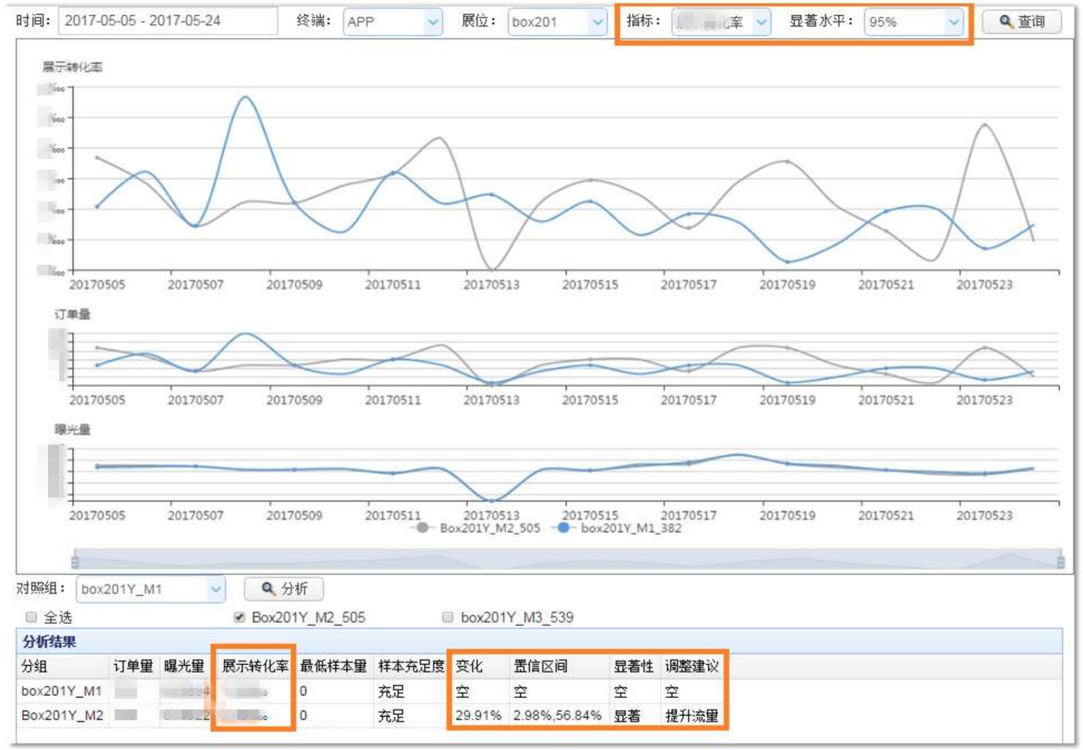
TABLE OF CONTENTES

TABLE OF CONTENTES

A/B Testing

A/B testing

利用分组的反馈指标对流量进行调整或重新配置



A/B Testing

A/B testing

关键点

01 保证其他因素不变的情况下，改变并观察单因素带来的变化，避免多个 A/B 测试特性的相互影响

02 样本少，统计量显著度不高

03 需要进行较长时间测试

04 尽量保证一部分“自然”流量

Exploitation & Exploration

Exploitation

基于已知最好策略，开发利用已知具有较高回报的item（贪婪、短期回报）
Pros：充分利用已知高回报item
Cons：陷于局部最优，错过潜在更高回报item的机会

Exploration

不考虑曾经的经验，勘探潜在可能高回报的item（非贪婪、长期回报）
Pros：发现更好回报的item
Cons：充分利用已有高回报item机会减少（如已经找到最好item）

目标

要找到Exploitation & Exploration的trade-off，以达到累计回报最大化

Exploitation & Exploration

E&E

ε-Greedy

选一个(0,1)之间较小的数ε，每次决策以概率ε去勘探Exploration，
1-ε的概率来开发Exploitation，基于选择的item及回报，更新item的回报期望

Pros：可以控制对Exploration和Exploitation的偏好程度

Cons:

设置最好的ε比较困难，大则适应变化较快，但长期累积回报低，小则适应变好的能力不够，但能获取更好的长期回报

策略运行一段时间后，我们对item的好坏了解的确定性增强，但仍然花费固定的精力去exploration，浪费本应该更多进行exploitation机会

Exploitation & Exploration

E&E

Upper Confidence Bound

UCB思想是乐观地面对不确定性，以item回报的置信上限作为回报预估值的一类算法，

其基本思想是：

（exploitation）我们对某个item尝试的次数越多，对该item回报估计的置信区间越窄、估计不确定性降低，那些均值更大的item倾向于被多次选择，这是算法保守的部分

（exploration）对某个item的尝试次数越少，置信区间越宽，不确定性较高，置信区间较宽的item倾向于被多次选择，这是算法激进的部分

Exploitation & Exploration

E&E

Upper Confidence Bound

UCB思想计算item期望的公式为：

$$\bar{x}_j + \sqrt{\frac{2lnN}{n_j}}$$

均值越大，标准差越小，被选中的概率会越来越大，起到了exploit的作用；
同时那些被选次数较少的item也会得到试验机会，起到了explore的作用。

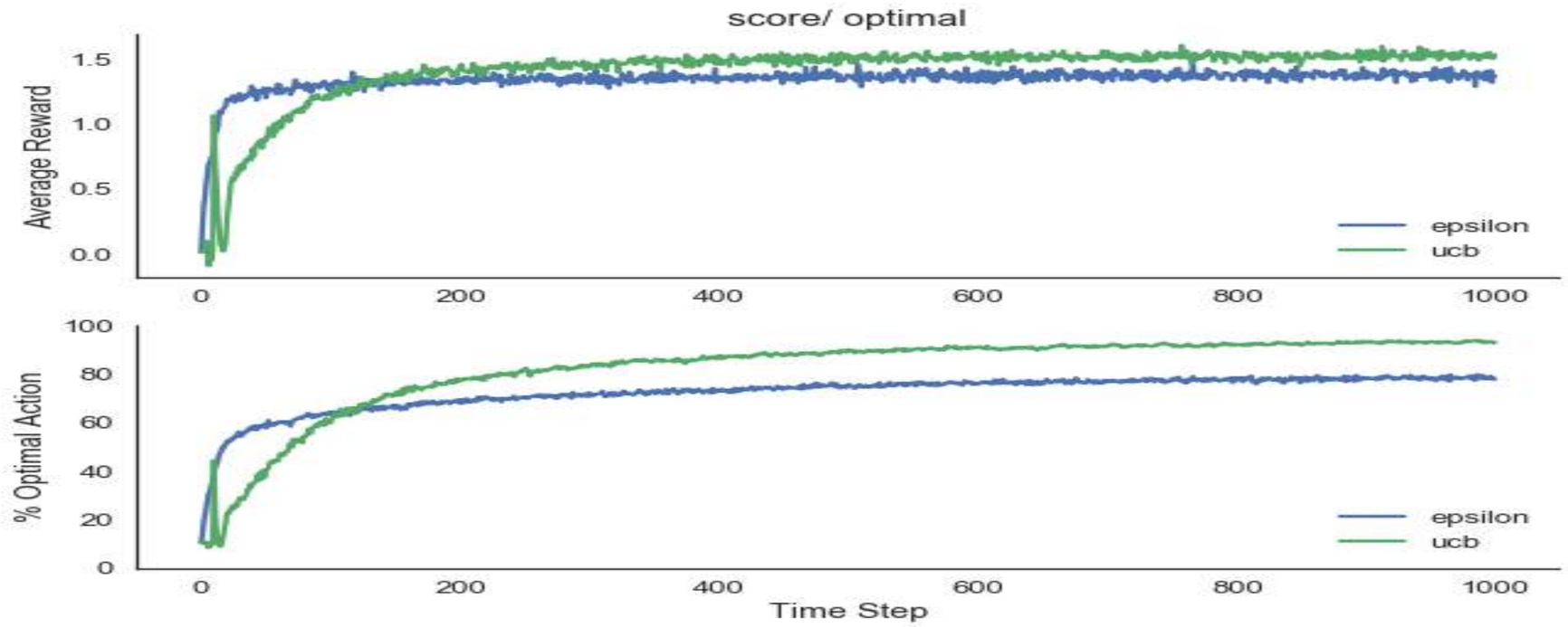
Pros： 考虑了回报均值的不确定性，让新的item更快得到尝试机会，将探索+开发融为一体，而且UCB算法不需要任何参数（vs epsilon-Greedy）

Cons： 需要首先尝试一遍所有item，一开始各item选择次数都比较少，导致得到的回报波动较大

Exploitation & Exploration

E&E

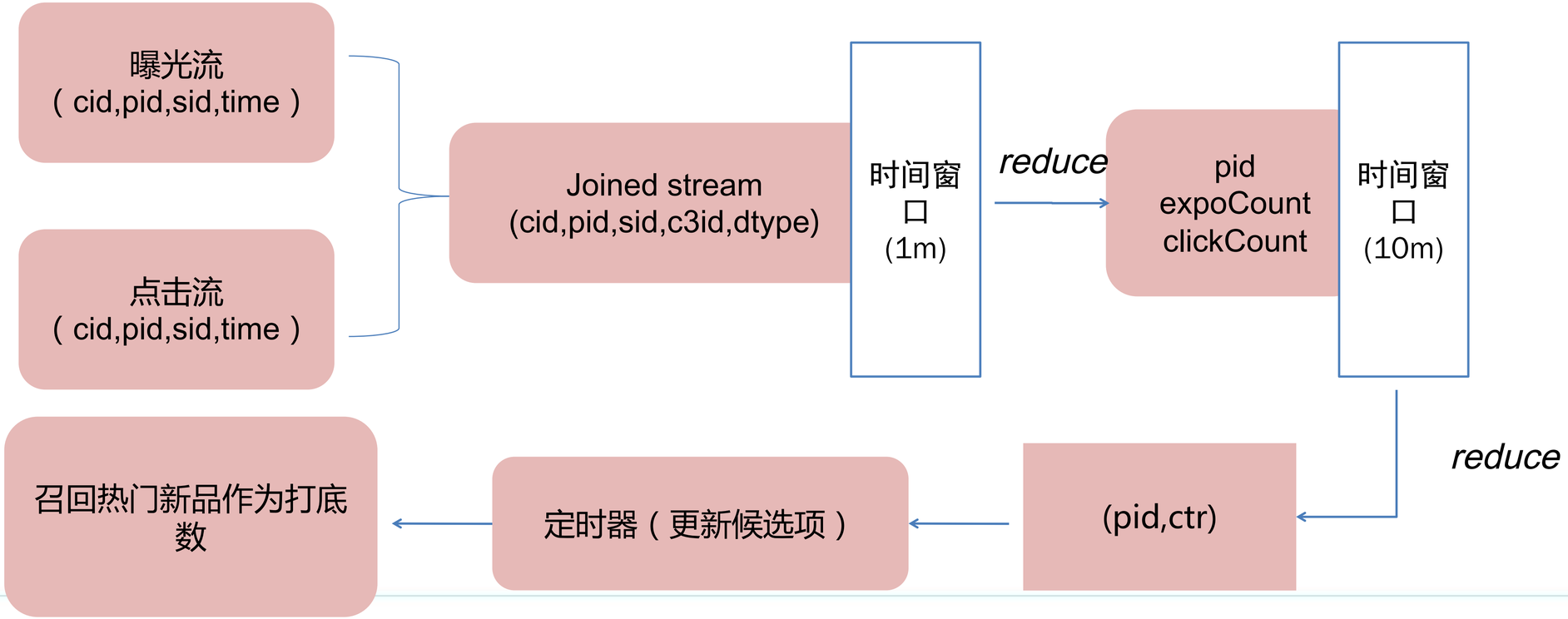
Epsilon VS UCB1



新品预测Flink-MAB

MAB

新品预测算法流程



新品预测Flink-MAB

```
  /**
    * 将每一条记录放进一个时间桶中，统计每个品类的行为或曝光数
    * @param dataSource
    * @return
    */
  def genWindowSlide(dataSource: DataStream[Line]): DataStream[(String,Int,Int)] ={
    val stream1 = dataSource.assignTimestampsAndWatermarks(new TimestampExtractor)
      .map(x => (x.productId,x.dtype,1))
      .keyBy(0)
      .window(TumblingEventTimeWindows.of(Time.seconds(15)))
//      .trigger(timeTrigger)
      .fold(("productId",0,0)){(acc,v) =>
        if(v._2.startsWith("click")){
          (v._1, acc._2+v._3, acc._3)
        }else
          (v._1, acc._2, acc._3+v._3)
      }.map { x =>
        try{
          (gcache.getC3id(x._1),x._2, x._3)
        }catch{case e: NullPointerException=>{
          e.printStackTrace()
          println("productId:"+x._1)
          ("c3id", -1, -1)
        }}
      }.filter(_._1 != "c3id")

//      .assignAscendingTimestamps(_ => System.currentTimeMillis())
    val stream2 = stream1.keyBy(0)
      .window(TumblingProcessingTimeWindows.of(Time.minutes(10)))
      //todo (bug) can't use rich function for windowStream
      .reduce((v1, v2) =>(v1._1,v1._2+v2._2, v1._3+v2._3))
    return stream2
  }
```

新品预测Flink-MAB

```scala
/**
  * join 两个或多个流
  * bug, surely; tried so many times; doesn't work when the left stream element can't match the right stream element
  * @param appBhvSet
  * @param exposureSet
  */
def joinStreams(appBhvSet:DataStream[(String,Int)], exposureSet:DataStream[(String,Int)]): Unit ={
  val joinedStream = appBhvSet.join(exposureSet)
  val outStream = joinedStream
    .where(_._1)
    .equalTo(_._1)
    .window(TumblingProcessingTimeWindows.of(Time.minutes(5)))
    .apply((v1, v2)=>(v1._1,v1._2+":"+v2._2))
  outStream.print()
}
//sink function
class MyTextSink[IN](format: TextOutputFormat[IN]) extends OutputFormatSinkFunction[IN](format: OutputFormat[IN]){
  override def open(parameters: Configuration): Unit = {
    super.open(parameters)
  }
  override def invoke(record: IN): Unit = {
    super.invoke(record)
  }
}
/**
  * event time extractor
  */
class TimestampExtractor extends AssignerWithPeriodicWatermarks[Line] with Serializable {
  override def getCurrentWatermark: Watermark =  {
    new Watermark(System.currentTimeMillis - 5000)
  }
  override def extractTimestamp(element: Line, previousElementTimestamp: Long): Long = {
    if(element.time == 0L)
      System.currentTimeMillis()
    else
      element.time
  }
}
```

新品预测Flink-MAB

```scala
/**
  * Created by GOME on 2017/12/22.
  * using UCB1 policy
  */
class Agent(armAr:Array[String]) extends Runnable{
  //摇出n个品类
  val targetArmNum: Int = 10
  val rewardPath = "D:\\flink-1.3.1\\log\\dd.txt"
  var lastChosen = new Array[Int](targetArmNum)

  val armNum: Int = armAr.length

  val c: Int = 2
  var expCount:Int = 0
  var armArray: Array[String] = armAr
  var armMap: Map[String,Int] = Map()
  var chosenCount = new Array[Int](armNum)
  var valueEstimate = new Array[Double](armNum)
  this.reset

  //每隔一段时间执行
  override def run(): Unit = {
    expCount += 1
    lastChosen = choose
    update

  }

  //重新设置value
  def reset(): Unit ={
    expCount = 0
    for(i <- 0 until armArray.length){
      armMap += (armArray(i) -> i)
    }
    chosenCount = new Array[Int](armNum)
    valueEstimate = new Array[Double](armNum)
  }
```

```scala
  //选择n台赌博机or品类
  def choose(): Array[Int] ={
    val tempArray = new Array[(Int,Double)](armNum)
    for(i <- 0 until tempArray.length){
      val tempD = valueEstimate(i) + math.sqrt(2 * scala.math.log(this.expCount + 1)/chosenCount(i))
      tempArray(i) = (i,if(tempD.isNaN) 0.0 else tempD)
    }
    val st = shuffle(tempArray).sortBy(_._2).reverse
    val rs = st.take(targetArmNum).map(x => x._1)
    println("choosing the best arm .........")
    println("the chosen arms are: %s  ".format(rs.mkString(",")))
    return rs
  }

  //更新结果
  def update(): Unit ={

    val arr = Agent.readReward(rewardPath)
    for(i <- 0 until arr.length){
      val ss = arr(i).split(",")
      val c3id = ss(0)
      val click = Integer.parseInt(ss(1))
      val expo = Integer.parseInt(ss(2))
      var reward:Double = 0.0
      if(expo != 0){
        reward = click*1.0/expo
      }

      val c3idIndex = armMap(c3id)
      if(lastChosen.contains(c3idIndex)){
        chosenCount(c3idIndex) += 1
        val g = 1.0/chosenCount(c3idIndex)
        valueEstimate(c3idIndex) += g*(reward - valueEstimate(c3idIndex))
        println("updating the reward ........")
        println("c3id is %s,  the index is %d, the valueestimate is %f".format(c3id, c3idIndex, valueEstimate(c3idIndex)))
      }}}}
```

新品预测Flink-MAB

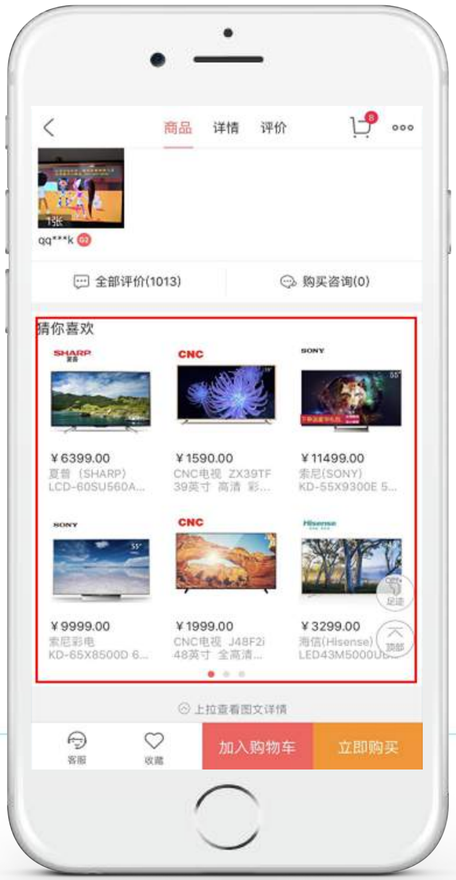
```
  /**
    * 将每一条记录放进一个时间桶中，统计每个品类的行为或曝光数
    * @param dataSource
    * @return
    */
  def genWindowSlide(dataSource: DataStream[Line]): DataStream[(String,Int,Int)] ={
    val stream1 = dataSource.assignTimestampsAndWatermarks(new TimestampExtractor)
      .map(x => (x.productId,x.dtype,1))
      .keyBy(0)
      .window(TumblingEventTimeWindows.of(Time.seconds(15)))
//      .trigger(timeTrigger)
      .fold(("productId",0,0)){(acc,v) =>
        if(v._2.startsWith("click")){
          (v._1, acc._2+v._3, acc._3)
        }else
          (v._1, acc._2, acc._3+v._3)
      }.map { x =>
        try{
          (gcache.getC3id(x._1),x._2, x._3)
        }catch{case e: NullPointerException=>{
          e.printStackTrace()
          println("productId:"+x._1)
          ("c3id", -1, -1)
        }}
      }.filter(_._1 != "c3id")

//      .assignAscendingTimestamps(_ => System.currentTimeMillis())
    val stream2 = stream1.keyBy(0)
      .window(TumblingProcessingTimeWindows.of(Time.minutes(10)))
      //todo (bug) can't use rich function for windowStream
      .reduce((v1, v2) =>(v1._1,v1._2+v2._2, v1._3+v2._3))
    return stream2
  }
```

新品预测Flink-MAB

MAB

新品预测算法流程



转化率提升：43.51%

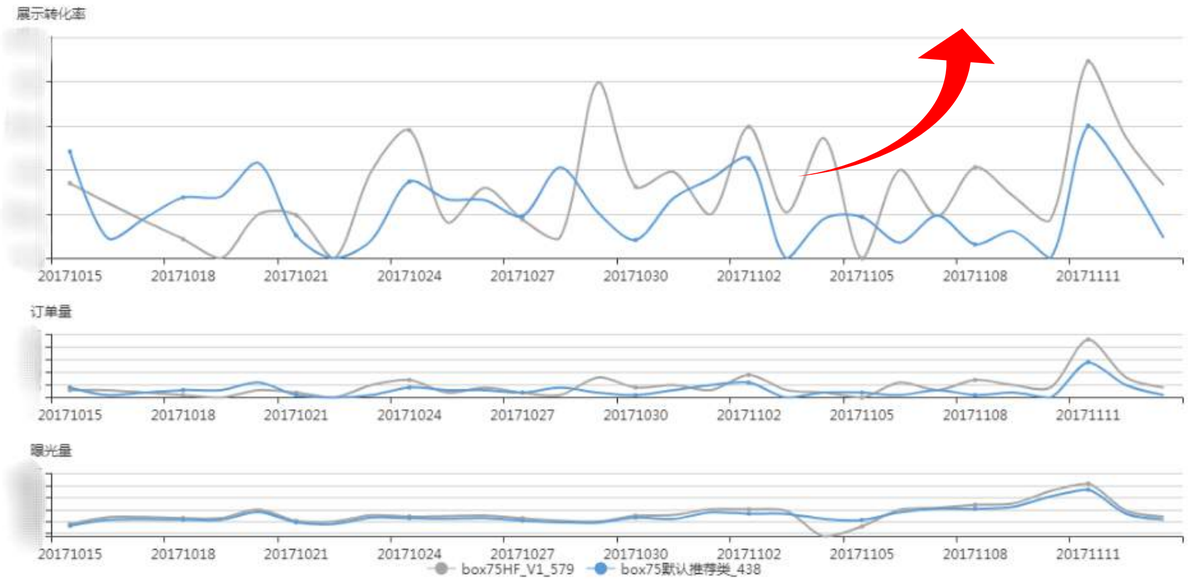
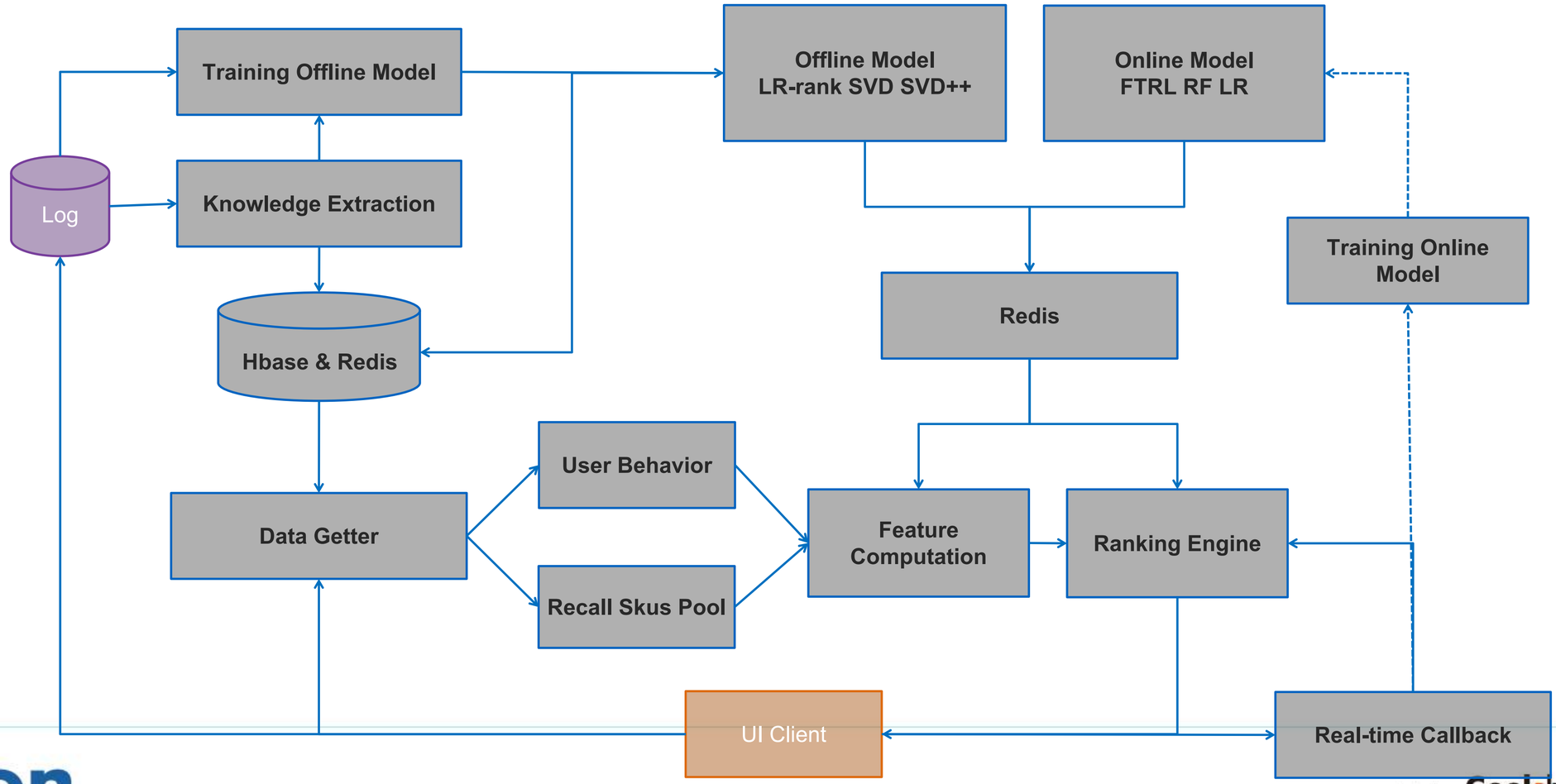


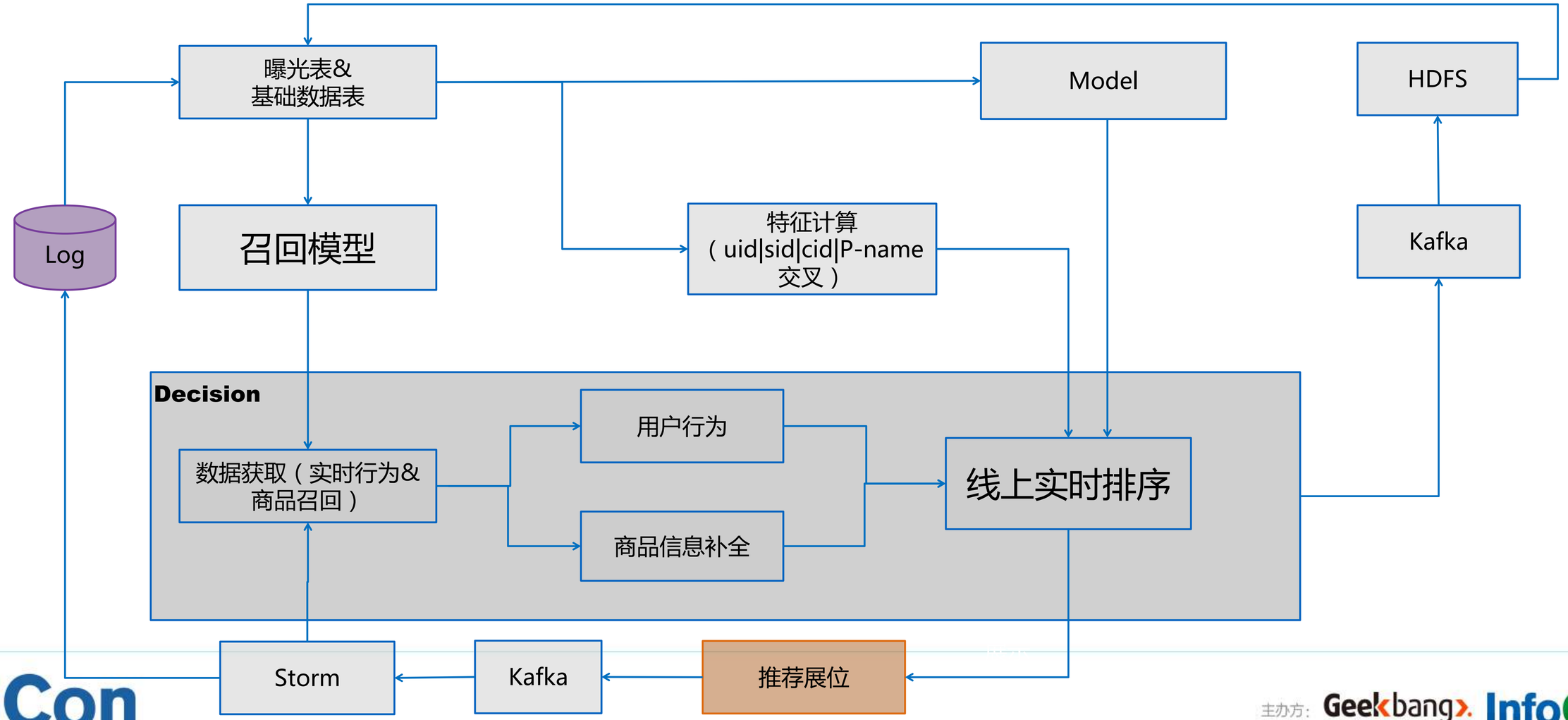
TABLE OF CONTENTES

TABLE OF CONTENTES

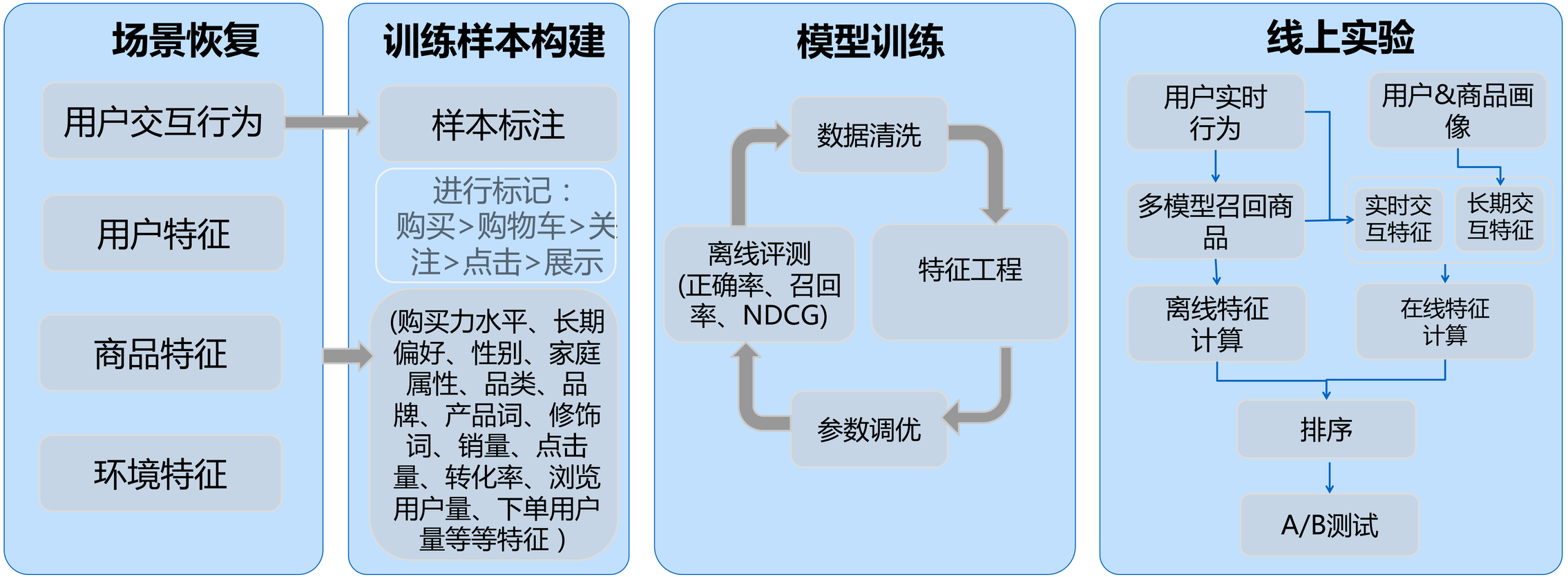
常见电商推荐架构


Training Offline Model
Offline Model
LR-rank SVD SVD++
Online Model
FTRL RF LR
Log
Knowledge Extraction
Training Online
Model
Redis
Hbase & Redis
User Behavior
Data Getter
Feature
Computation
Ranking Engine
Recall Skus Pool
UI Client
Real-time Callback

通用L2R Pipeline



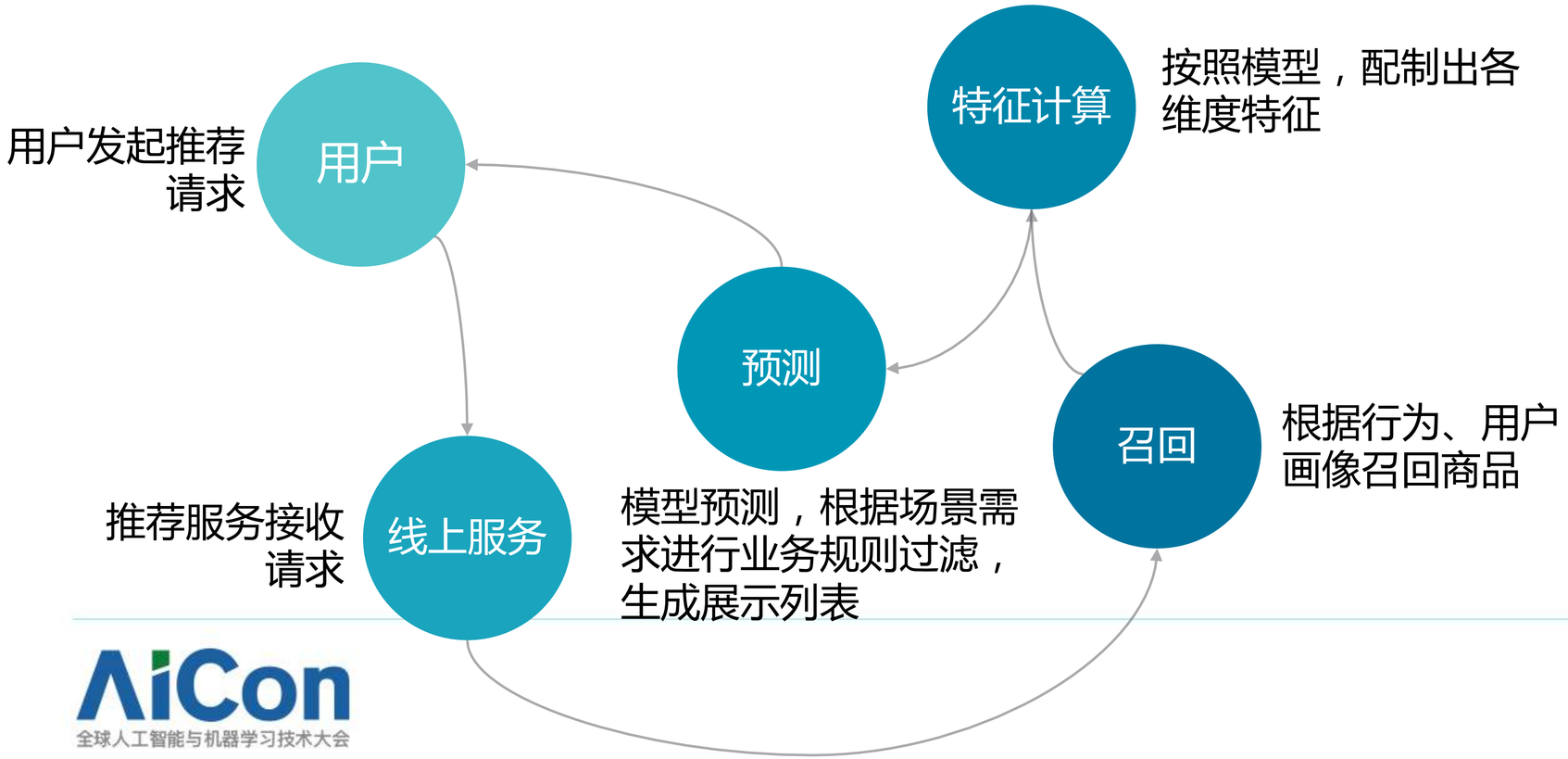
实时排序部署流程


场景恢复
用户交互行为
用户特征
商品特征
环境特征
训练样本构建
样本标注
进行标记：
购买>购物车>关注>点击>展示
(购买力水平、长期偏好、性别、家庭属性、品类、品牌、产品词、修饰词、销量、点击量、转化率、浏览用户量、下单用户量等等特征）
模型训练
数据清洗
特征工程
参数调优
离线评测
(正确率、召回率、NDCG)
线上实验
用户实时行为
用户&商品画像
多模型召回商品
实时交互特征
长期交互特征
离线特征计算
在线特征计算
排序
A/B测试

有关通用架构的思考

架构

Pros & Cons



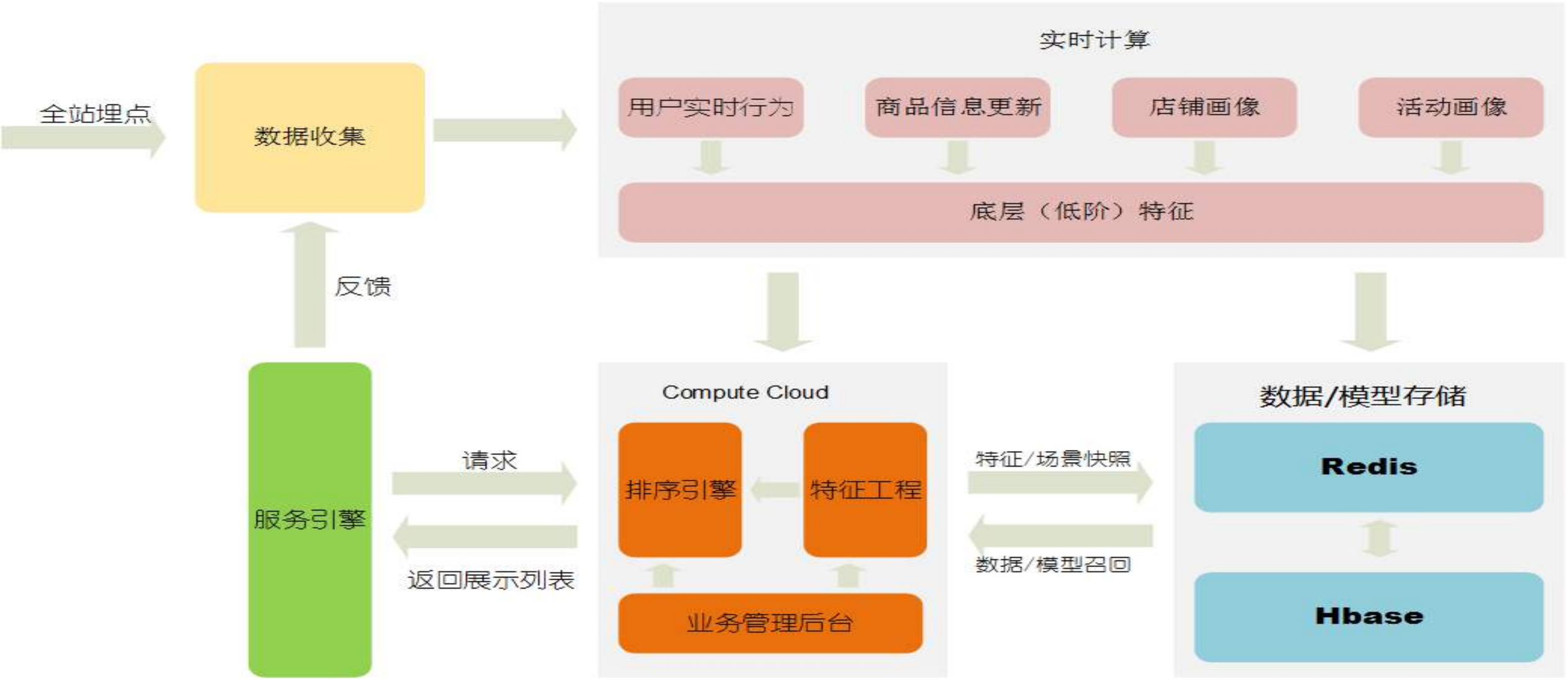
优势

逻辑简单，符合机器学习常用的workflow

不足

工程师线上&线下代码逻辑很难保证一致
离线训练模型需要恢复场景
延时较长，可能会影响体验
线上服务大多数情况下资源利用率低

优化方案


全站埋点
数据收集
反馈
实时计算
用户实时行为
商品信息更新
店铺画像
活动画像
底层（低阶）特征
服务引擎
请求
返回展示列表
Compute Cloud
排序引擎
特征工程
业务管理后台
特征/场景快照
数据/模型召回
数据/模型存储
Redis
Hbase

触发流程

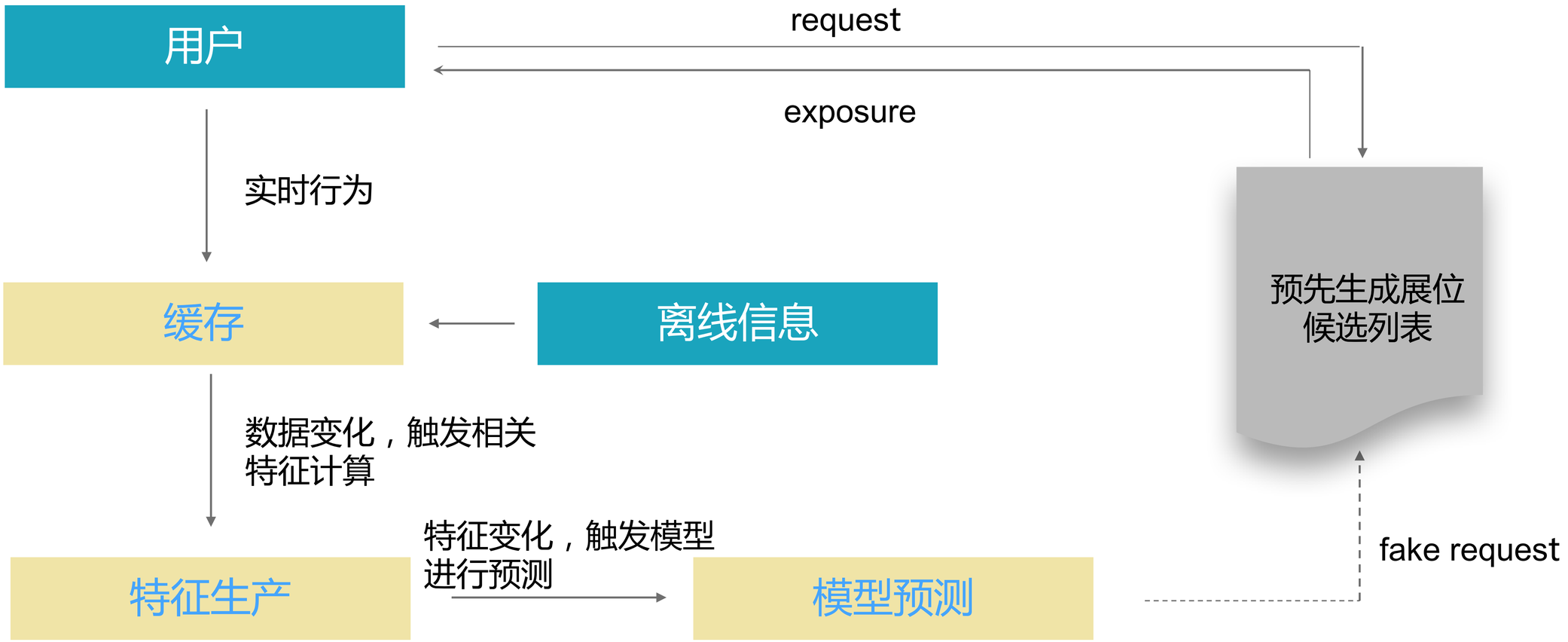

用户
request
exposure
实时行为
缓存
离线信息
预先生成展位
候选列表
数据变化，触发相关
特征计算
特征变化，触发模型
进行预测
特征生产
模型预测
fake request

TABLE OF CONTENTES

TABLE OF CONTENTES

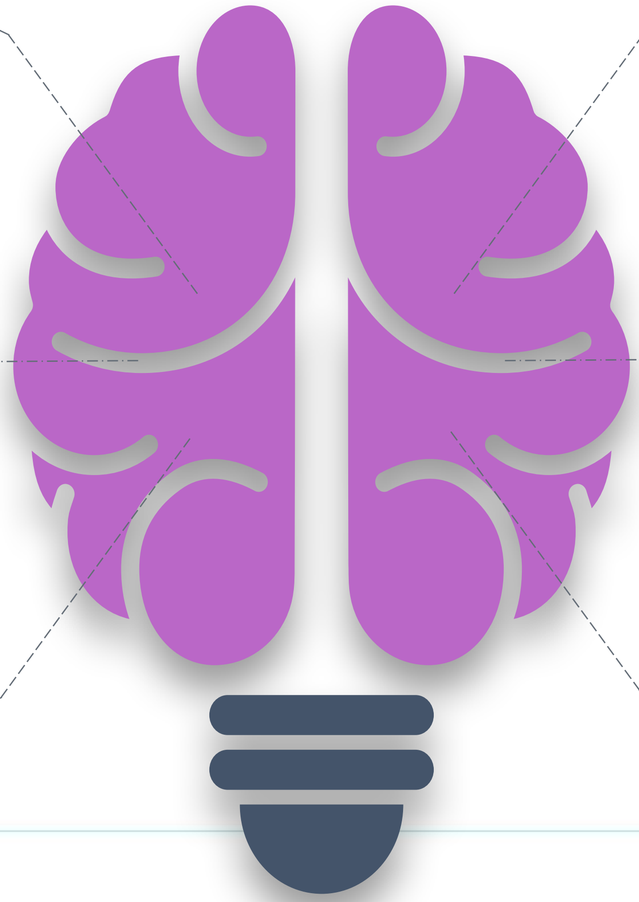
总结

召回和排序没有轻重之分

排序之路

人工 → 机器学习 → 深度学习

架构瓶颈



召回方法创新

融入视觉特征的排序

强化学习

Thanks!