一个推荐系统的实现

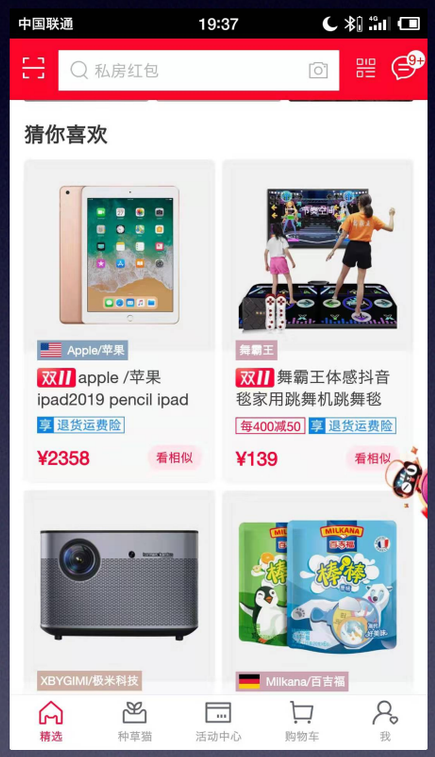
美篇算法组-杨瑞

推荐系统概述

推荐系统算法

推荐系统架构

无处不在的推荐



推荐系统几乎成为一个互联网产品的标配。我们每天都在跟各种各样的推荐系统打交道。

推荐场景的分类

- 相关推荐：“相关推荐”，“你可能还喜欢”，“猜你喜欢”等
- 热门推荐：“热门”，“爆款”等
- 个性化推荐：“发现”，“推荐”，“首页”等

美篇的推荐场景

个性化推荐

相关推荐

热门推荐

推荐系统是什么

A recommender system is a subclass of information filtering system that seeks to predict the "rating" or “preference” a user would give to an item.

——wiki

推荐系统是一种信息过滤系统，用于预测用户对物品的“评分”或“偏好”。

- 信息过滤： 这是一个信息过载（information overload）的时代。信息的生产者和消费者都遇到了极大的挑战。人们需要从大量信息中获取自己感兴趣的信息。推荐系统由此诞生。

- 预测用户对物品的偏好： 一个好的推荐系统能把根据物品的特征，用户的历史行为准确评估用户对物品的偏好，推荐特定的物品给特定的用户，从而达到信息生产者和消费者的双赢。

形式化表达

进一步用形式化的语言解释，推荐系统其实就是在拟合一个用户对物品满意度的函数：

$$y = f\left(x_u, x_c, x_i\right)$$

x_u	用户特征
x_c	环境特征
x_i	物品特征
y	用户满意度

推荐系统算法篇

基于内容的推荐

- 关注内容本身的特征
- 核心在于物品表示（Item Representation） 及物品间的相似性度量
- 有了物品间的相似性度量，就可以根据过去喜欢的物品，推荐与之相似的物品
- 基于内容的推荐，重要的不是推荐算法，而是内容挖掘和分析。

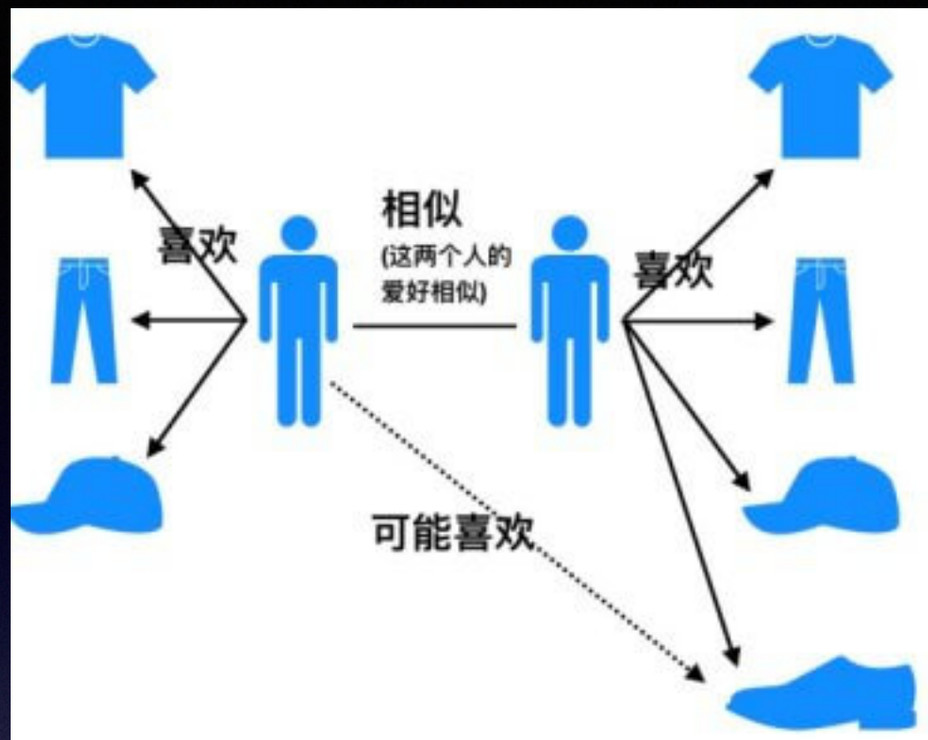
用户画像


首页 / 个人画像
首页 ×
个人画像 ×
30315960
查询
阅读偏好
时事-突发事件
历史-唐朝历史
文学-名著赏析
摄影-摄影教程
娱乐-影评
文学-名人古诗
历史-明朝历史
生活-生活随记
文学-诗社合集
历史-老照片故事
情感-爱情
摄影-手机摄影
文学-读后感
历史-清朝历史
历史-历史人物
美食-甜品
情感-回忆
历史-历史解密
军事-军队文化
历史-南北朝
历史-帝王
历史-近现代战争
军事-大国博弈
生活-玩乐
军事-当代战争
美食-主食
文学-散文
美食-各地小吃
娱乐-剧评
历史-历史研究
历史-历史事件
历史-考古
情感-故乡情
画会-艺术
美食-食材
健康-养生
生活-园艺
文学-红楼梦
军事-军事新闻
艺术-书法
娱乐-电影
情感-亲情
文学-唐诗
摄影-静物摄影
文学-名人现代诗
文学-心情日记
娱乐-电视剧
生活-萌娃
摄影-风光摄影
旅行-大洋洲
健康-医疗疾病

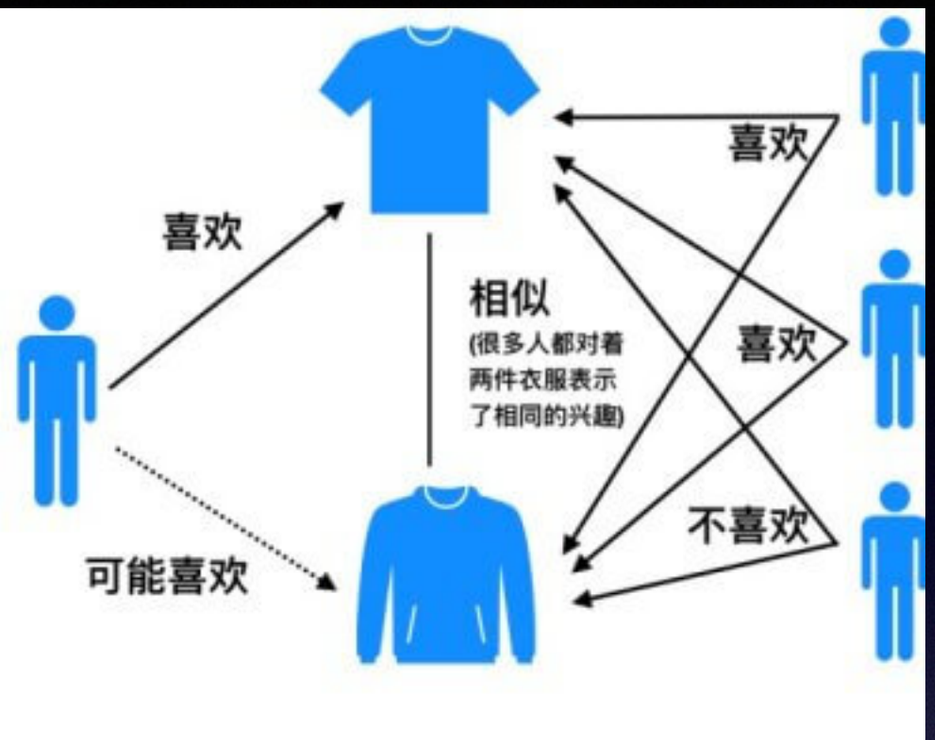
协同过滤

“物以类聚，人以群分。”

—《战国策》



“人以群分”的基于用户的协同过滤



“物以类聚”的基于物品的协同过滤

两个假设：

- 喜欢相同物品列表的用户具备某种相同的特质：

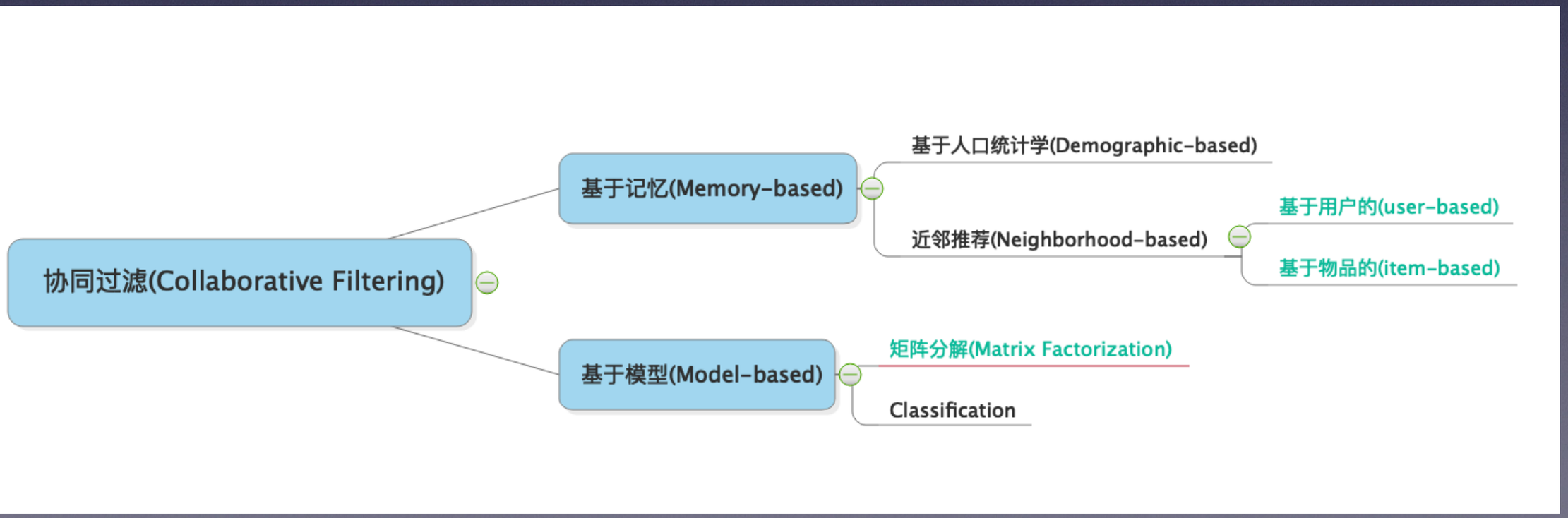
  推荐与他相似的用户喜欢的物品

- 被相同用户群体喜欢的物品具备某种相同的特质：

  推荐与他之前喜欢物品相似的物品

协同过滤

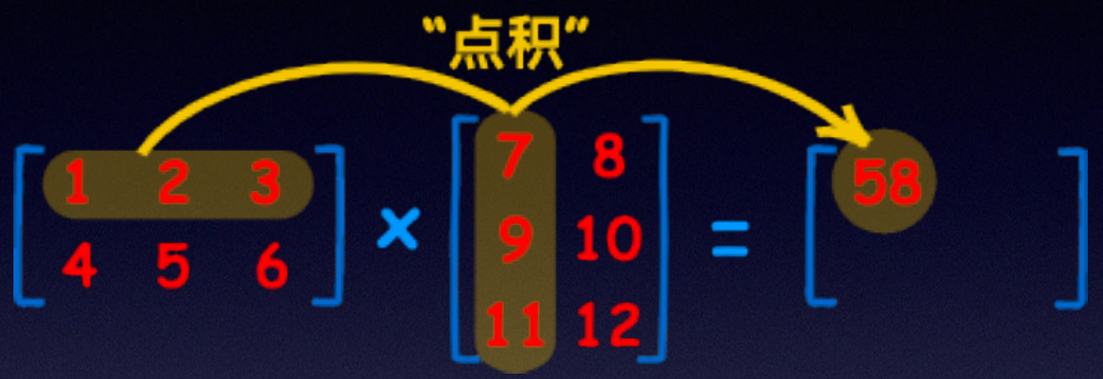
- 协同过滤通过收集和分析大规模的用户行为来预测用户对物品的偏好
- 关注协同特征而不是内容本身。
- 花样繁多，广泛应用



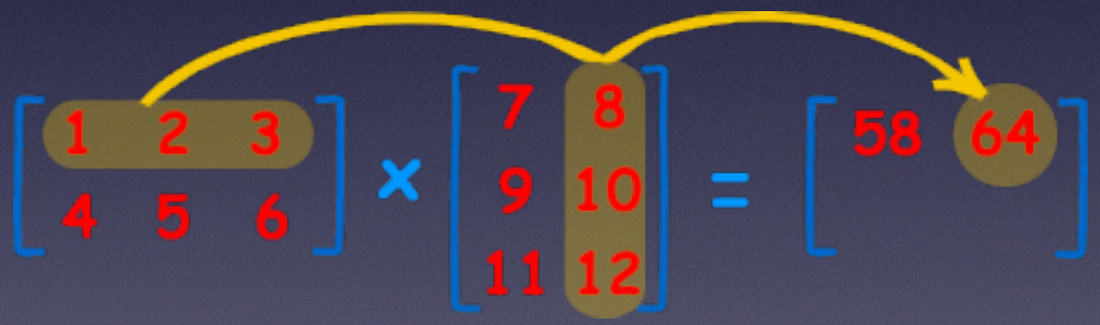
矩阵分解

隐语义，机器学习，协同过滤

矩阵乘法——一个例子



$(1, 2, 3) \bullet (7, 9, 11) = 1\times7 + 2\times9 + 3\times11 = 58$



$(1, 2, 3) \bullet (8, 10, 12) = 1\times8 + 2\times10 + 3\times12 = 64$

. . . . . .

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

✓

用户评分矩阵

我们收集的用户行为经常是这样的：
用户 user1 点击了文章 item1；用户 user2 观看了视频 item2 10min；用户 user3 给电影 item3 评3.5分……
它们或隐式，或显式，都是在表达用户给物品的评分。

	item1	item2	item3	item4	item5
f1	3	1	1	3	1
f2	1	2	4	1	3

	f1	f2
user1	1	0
user2	0	1
user3	1	0
user4	1	1

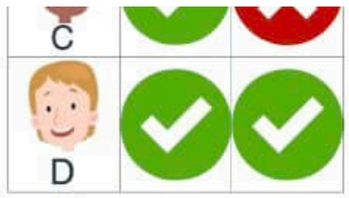
	item1	item2	item3	item4	item5
user1	3	1	1	3	?
user2	1	2	?	1	3
user3	3	?	1	3	1
user4	4	3	?	4	4

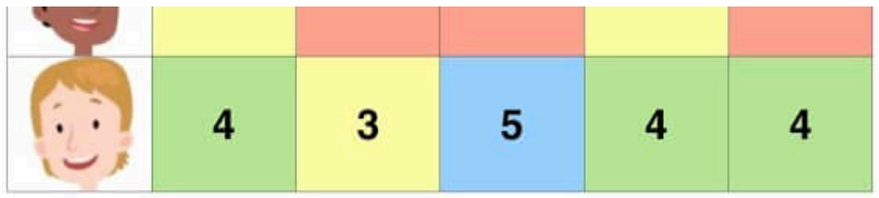
这样做有道理吗?

	item1	item2	item3	item4	item5
f1	3	1	1	3	1
f2	1	2	4	1	3

	f1	f2
user1	1	0
user2	0	1
user3	1	0
user4	1	1

	item1	item2	item3	item4	item5
user1	3	1	1	3	?
user2	1	2	?	1	3
user3	3	?	1	3	1
user4	4	3	?	4	4



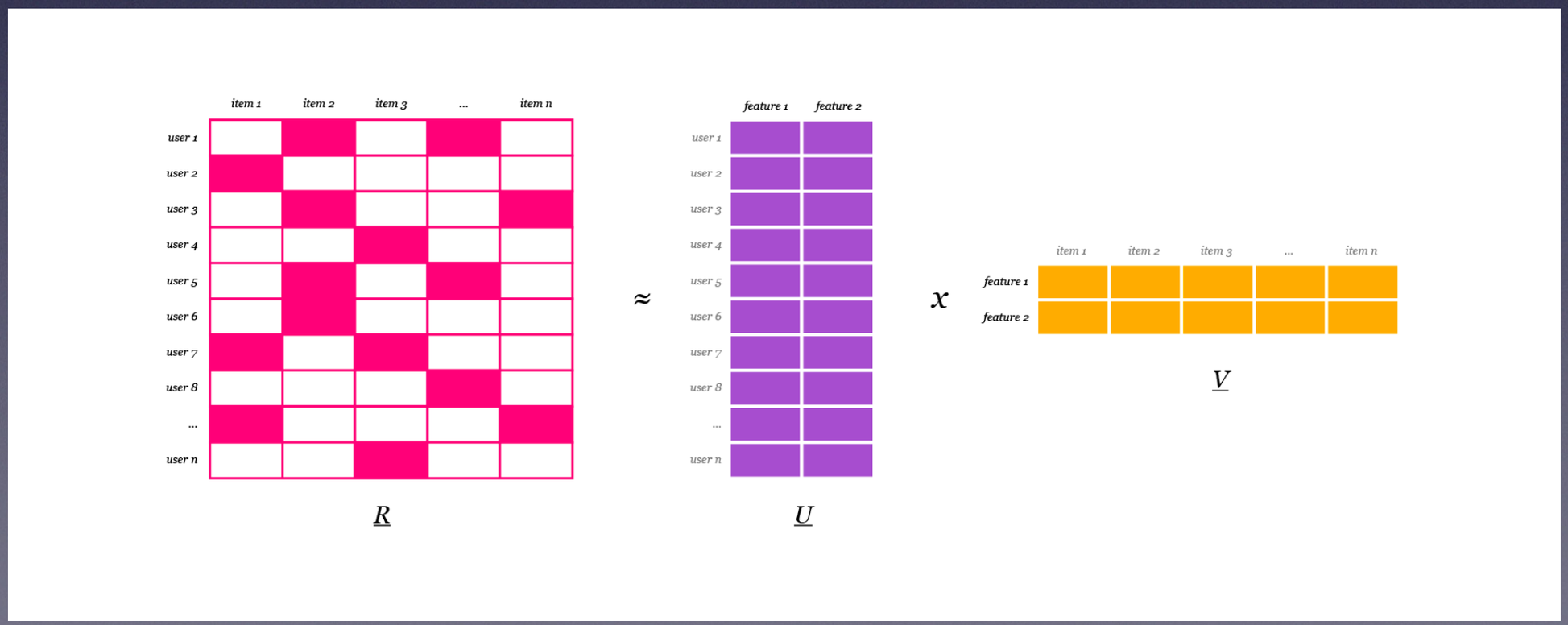


矩阵分解

- 矩阵分解将一个大矩阵分解为两个小矩阵（矩阵降维）。其中一个是用户-隐因子矩阵，另一个是物品-隐因子矩阵。

- 矩阵分解属于协同过滤算法，也体现了“物以类聚人以群分”的集体智慧。

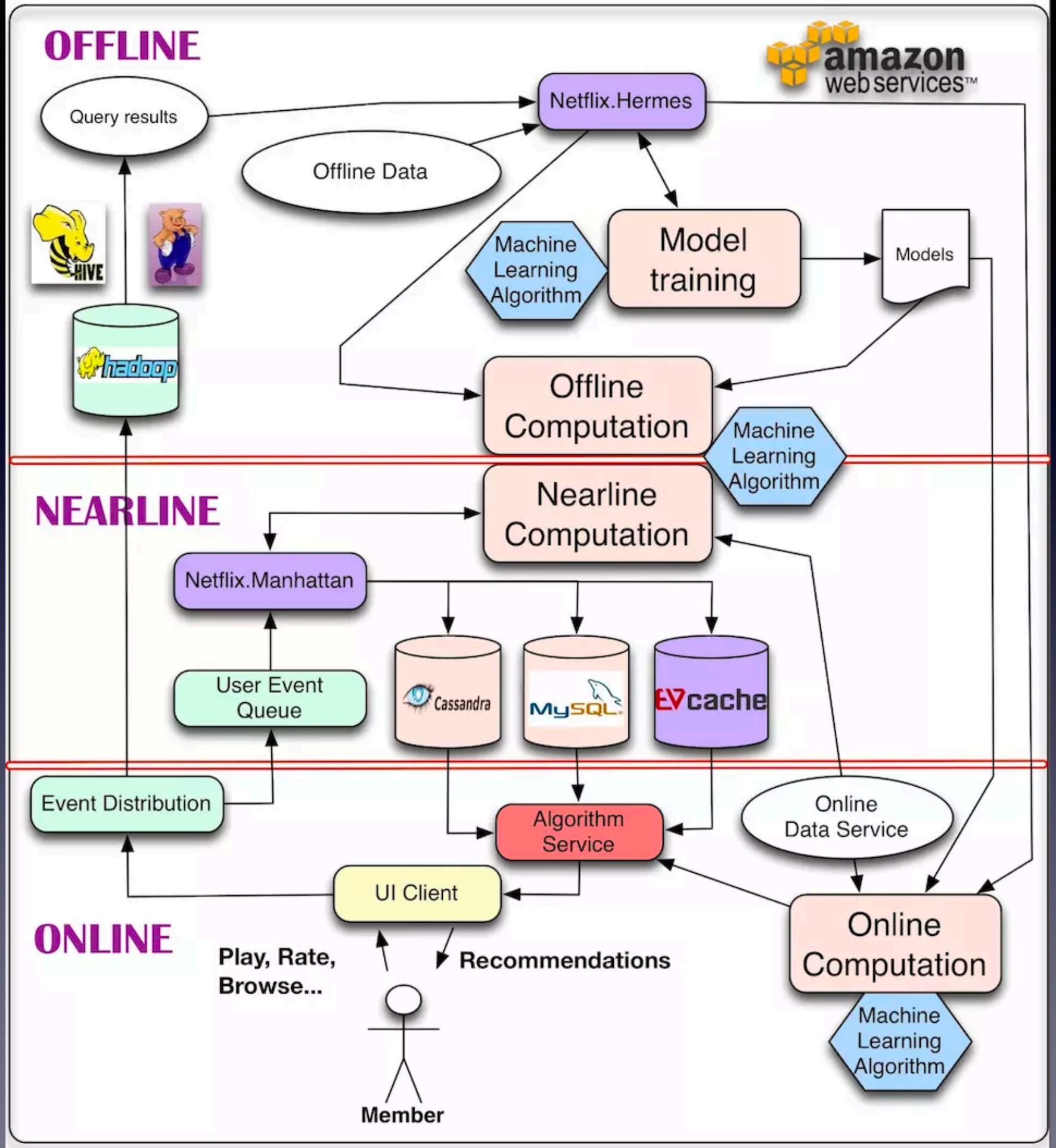
- 矩阵分解表达式：$\mathrm{R}_{m\times n} \approx U_{m\times k} \cdot V^{T}_{n\times k}$



Spark MLlib 实现一个简单的基于矩阵分解的推荐系统

```
from pyspark import SparkContext
from pyspark.mllib.recommendation import ALS

sc = SparkContext()
sc.setCheckpointDir('/Users/forrest/Documents/data/checkpoint/')
ratings_file = '/Users/forrest/Downloads/data/user_movies_desensitization.csv'

ratings_raw_data = sc.textFile(ratings_file)
ratings_data = ratings_raw_data.map(lambda line: line.split(",")).map(lambda tokens: (token
s[0], tokens[1], tokens[2])).cache()

training_RDD, validation_RDD, test_RDD = ratings_data.randomSplit([6, 2, 2], seed=0)
validation_for_predict_RDD = validation_RDD.map(lambda x: (x[0], x[1]))
test_for_predict_RDD = test_RDD.map(lambda x: (x[0], x[1]))

seed = 5
best_iterations = 20
best_lmda = 0.2
best_rank = 15

# 训练模型
model = ALS.train(training_RDD, best_rank, seed=seed, iterations=best_iterations, lambda_=b
est_lmda)
# 给user推荐num个product
print(model.recommendProducts(88, 10))
sc.stop()
```

推荐系统架构篇


OFFLINE
amazon web services™
Query results
Netflix.Hermes
Offline Data
Machine Learning Algorithm
Model training
Models
hadoop
Offline Computation
Machine Learning Algorithm
NEARLINE
Nearline Computation
Netflix.Manhattan
User Event Queue
Cassandra
MySQL
EVcache
Event Distribution
Algorithm Service
Online Data Service
UI Client
ONLINE
Play, Rate, Browse...
Recommendations
Online Computation
Machine Learning Algorithm
Member

Netflix 架构

三层计算架构

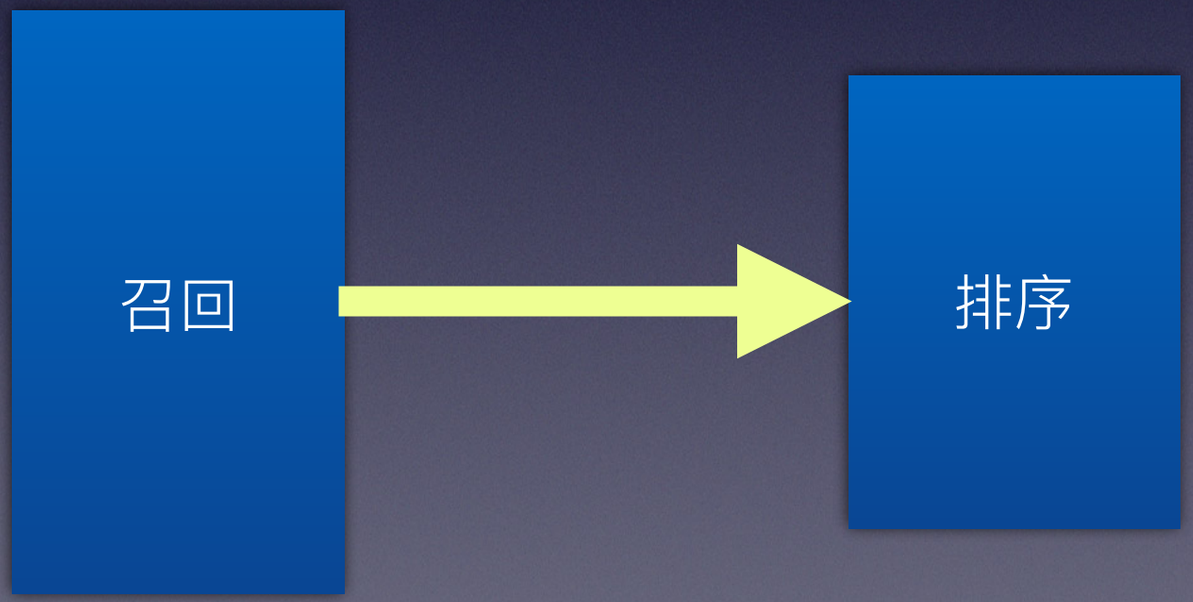
- 在线层： 实时特征，快速响应，服务高可用
- 离线层：规模计算，复杂模型
- 近线层：折中方案，异步进行，增量学习

在线层-Feed服务

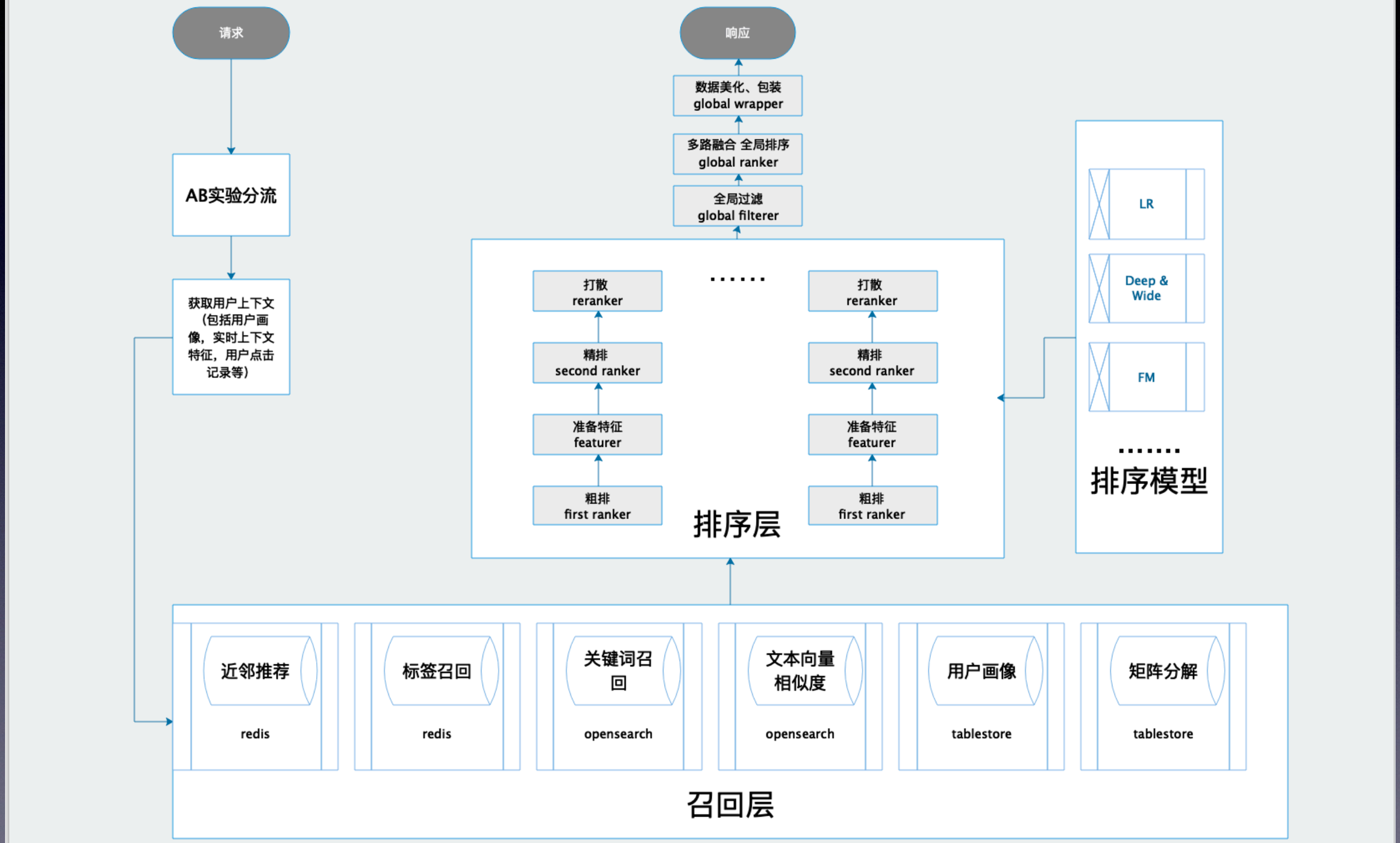
推荐流程一般分为召回和排序两个阶段。

召回层：从全量内容池里筛选出用户可能喜欢的内容候选集。

排序层：对召回的候选集进行打分排序，选出最高的几个结果。



美篇推荐系统架构


请求
AB实验分流
获取用户上下文（包括用户画像，实时上下文特征，用户点击记录等）
响应
数据美化、包装
global wrapper
多路融合 全局排序
global ranker
全局过滤
global filterer
打散
reranker
精排
second ranker
准备特征
featurer
粗排
first ranker
.....
打散
reranker
精排
second ranker
准备特征
featurer
粗排
first ranker
排序层
LR
Deep & Wide
FM
.....
排序模型
近邻推荐
redis
标签召回
redis
关键词召回
opensearch
文本向量相似度
opensearch
用户画像
tablestore
矩阵分解
tablestore
召回层

```
{
  "name": "app_home_feed",
  "global_filter": "unique_item_filter",
  "global_ranker": "app_home_global_ranker_20190916_1003",
  "global_wrapper": "app_home_wrapper_exp_20191016",
  "count": 15,
  "experiment_id": "",
  "activity_stream": [{
    "name": "alg_wheel_1",
    "type": "alg_wheel",
    "item_type": 1,
    "count": 15,
    "cache_prefix": "alg_wheel:1",
    "recaller": [{
      "name": "article_icf_recaller"
    }, {
      "name": "article_content_based_recaller_v2"
    }, {
      "name": "mf_article_feed_recaller",
      "count": 100
    }],
    "filter": [],
    "featurer": [{
      "name": "app_home_article_featurer"
    }],
    "first_ranker": [{
      "name": "app_home_first_ranker_exp_20191113_mf_u2u"
    }],
    "second_ranker": [{
      "name": "article_app_home_ctr_ranker_1004"
    }],
    "re_ranker": [{
      "name": "article_app_home_reranker"
    }]
  }]
}
```

策略热插拔配置

多路融合：多个车轮一起滚动

```
51      // multi activity stream
52      sp.Annotate(time.Now(), "Do fetch multi activity stream")
53      for _, wheelConfigJson := range conf.ActivityStreamConfigJson {
54          // fetch wheel config => wlc
55          wlc, _ := GetWheelConfigFromJson(req, &wheelConfigJson)
56          // fetch wheel => wl
57          wi, _ := engine.Engine.New(wlc.Type)
58          if wl, ok := wi.(wheel.Wheel); ok {
59              count++
60              // Do wheel roll
61              go func(_wl wheel.Wheel, _wlc *wheel.WheelConfig) {
62                  ch <- _wl.Roll(&newCtx, req, _wlc, showedArticleIds)
63              }(wl, &wlc)
64          } else {
65              log.Err(wheelConfigJson.Name + " wheel convert failed !")
66              continue
67          }
68      }
```

召回-排序：每个车轮如何滚动

```
52      //Do recall
53      recalledFeedCells := alg.AddMultiRecaller(ctx, req, showedItemIds, conf.Recaller...)
54      //Do filter
55      filterdFeedCells := alg.AddMultiFilter(ctx, req, recalledFeedCells, conf.Filter...)
56      //Do first ranker
57      firstRankedFeedCells := alg.AddMultiRanker(ctx, req, filterdFeedCells, conf.FirstRanker...)
58      //Do featurer
59      featuredFeedCells := alg.AddMultiFeaturer(ctx, req, firstRankedFeedCells, conf.Featurer...)
60      //Do second ranker
61      secondRankedFeedCells := alg.AddMultiRanker(ctx, req, featuredFeedCells, conf.SecondRanker...)
62      //Do re reanker
63          rerankedCells := alg.AddMultiRanker(ctx, req, secondRankedFeedCells, conf.ReRanker...)
```

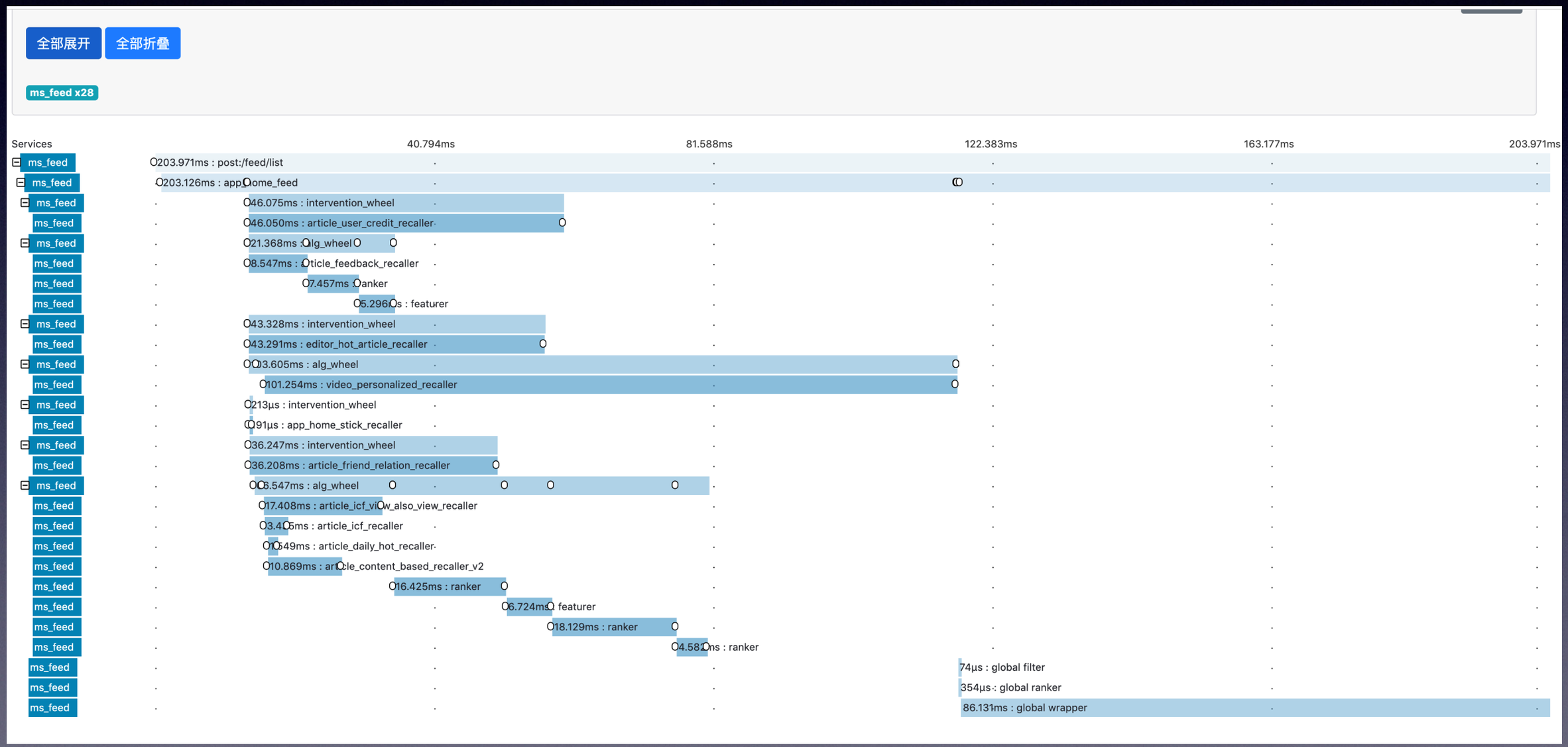
多路召回

```
100 // 多路召回
101 func (alg AlgWheel) AddMultiRecaller(ctx *context.Context, req *model.FeedRequest, showedItemIds []int64, rs ...RecallerConfig) (feedCells []model.FeedCell) {
102     var ch = make(chan []model.FeedCell)
103     for _, r := range rs {
104         go func(fr recaller.Recaller, amount int, name string) {
105             feedCells := fr.Recall(ctx, req, amount, showedItemIds)
106             ch <- feedCells
107         }(r.Recaller, r.Count, r.Name)
108     }
109     for i := 0; i < len(rs); i++ {
110         feedCells = append(feedCells, <-ch...)
111     }
112     return
113 }
```

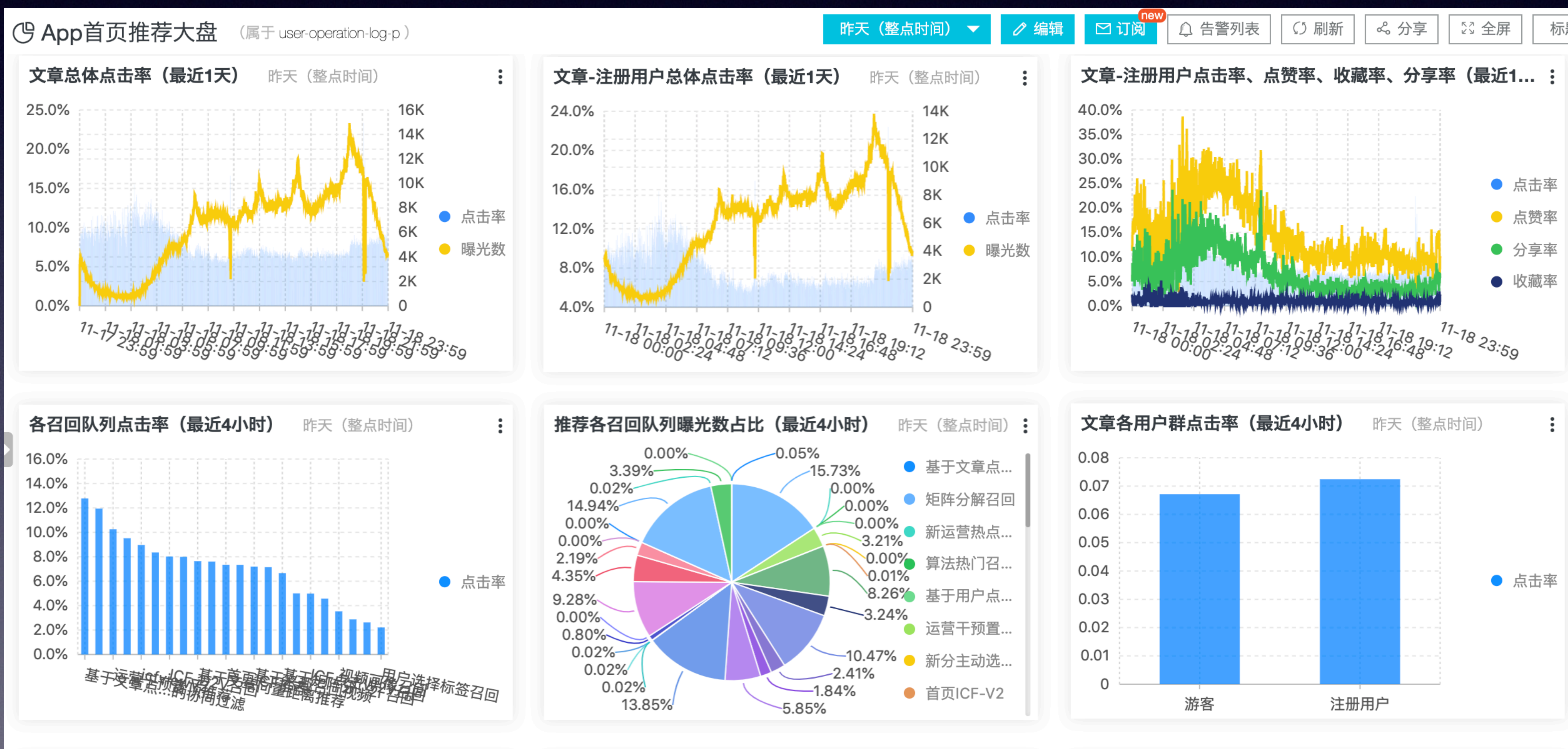
灵活实验配置

```
1 ADD | RECALLER | {"name": "alg_wheel_1", "recaller": {"name": "ucf_via_mf_user_vector_article_feed_recaller", "count": 200}}&
2 SET | WHEEL | {"name": "alg_wheel_1", "first_ranker": []}&
3 ADD | FIRST_RANKER | {"name": "alg_wheel_1", "first_ranker": {"name": "app_home_first_ranker_exp_20191113_mf_u2u"}}
```

全链路监控


全部展开
全部折叠
ms_feed x28
Services
40.794ms
81.588ms
122.383ms
163.177ms
203.971ms
ms_feed
203.971ms : post:/feed/list
ms_feed
203.126ms : app_home_feed
ms_feed
46.075ms : intervention_wheel
ms_feed
46.050ms : article_user_credit_recaller
ms_feed
21.368ms : alg_wheel
ms_feed
8.547ms : article_feedback_recaller
ms_feed
7.457ms : ranker
ms_feed
5.296ms : featurer
ms_feed
43.328ms : intervention_wheel
ms_feed
43.291ms : editor_hot_article_recaller
ms_feed
3.605ms : alg_wheel
ms_feed
101.254ms : video_personalized_recaller
ms_feed
213µs : intervention_wheel
ms_feed
91µs : app_home_stick_recaller
ms_feed
36.247ms : intervention_wheel
ms_feed
36.208ms : article_friend_relation_recaller
ms_feed
6.547ms : alg_wheel
ms_feed
17.408ms : article_icf_view_also_view_recaller
ms_feed
3.405ms : article_icf_recaller
ms_feed
549ms : article_daily_hot_recaller
ms_feed
10.869ms : article_content_based_recaller_v2
ms_feed
16.425ms : ranker
ms_feed
6.724ms : featurer
ms_feed
18.129ms : ranker
ms_feed
4.582ms : ranker
ms_feed
74µs : global filter
ms_feed
354µs : global ranker
ms_feed
86.131ms : global wrapper

业务监控



App首页推荐大盘 （属于 user-operation-log-p ）
昨天（整点时间）
编辑
订阅
new
告警列表
刷新
分享
全屏
标题
文章总体点击率（最近1天） 昨天（整点时间）
点击率
曝光数
文章-注册用户总体点击率（最近1天） 昨天（整点时间）
点击率
曝光数
11-18 23:59
文章-注册用户点击率、点赞率、收藏率、分享率（最近1...
点击率
点赞率
分享率
收藏率
11-18 23:59
各召回队列点击率（最近4小时） 昨天（整点时间）
点击率
推荐各召回队列曝光数占比（最近4小时） 昨天（整点时间）
基于文章点...
矩阵分解召回
新运营热点...
算法热门召...
基于用户点...
运营干预置...
新分主动选...
首页ICF-V2
文章各用户群点击率（最近4小时） 昨天（整点时间）
点击率
游客
注册用户

最后