协同过滤

推荐系统介绍

当下，个性化推荐成了互联网产品的标配。

推荐系统从搜索引擎借鉴了不少技术和思想，比如内容推荐有不少技术就来自搜索引擎，由 Amazon 发扬光大的。推荐系统也是现在热门的人工智能分支之一，未来随着各种硬件设备越来越智能，万物互联得越来越紧密，人们的个性化需求、场景的多样性、数据的复杂性都对推荐系统提出了更高的要求。

推荐系统概括一下，其实就是以下的的目标主要包括：

1. 用户满意性：首当其冲的，推荐系统主要就是为了满足用户的需求，因此准确率是评判一个推荐系统好坏的最关键指标。
2. 多样性：虽然推荐系统最主要还是满足用户的兴趣，但是也要兼顾内容的多样性，对于权重不同的兴趣都要做到兼顾。
3. 新颖性：用户看到的内容是那些他们之前没有听说过的物品。简单的做法就是在推荐列表去掉用户之前有过行为的那些内容。
4. 惊喜度：和新颖性类似，但新颖性只是用户没看到过的但是确实是和他行为是相关的，而惊喜度是用户既没有看过和他之前的行为也不相关，但用户看到后的确是喜欢的。
5. 实时性：推荐系统要根据用户的上下文来实时更新推荐内容，用户的兴趣也是随着时间而改变的，需要实时更新。
6. 推荐透明度：对于用户看到的最终结果，要让用户知道推荐此内容的原因。比如，“买过这本书的人同时也买过”、”你购买过的 xx 和此商品类似”。

7. 覆盖率：挖掘长尾内容也是推荐系统很重要的目标。因此，推荐的内容覆盖到的内容越多越好。

基于这些目标，推荐系统包括四种推荐方式：

1. 热门推荐：就是热门排行榜的概念。这种推荐方式不仅仅在 IT 系统，在平常的生活中也是处处存在的。这应该是效果最好的一种推荐方式，毕竟热门推荐的物品都是位于曝光量比较高的位置的。

2. 人工推荐：人工干预的推荐内容。相比于依赖热门和算法来进行推荐。一些热点时事如世界杯、nba 总决赛等就需要人工加入推荐列表。另一方面，热点新闻带来的推荐效果也是很高的。

3. 相关推荐：相关推荐有点类似于关联规则的个性化推荐，就是在你阅读一个内容的时候，会提示你阅读与此相关的内容。

4. 个性化推荐：基于用户的历史行为做出的内容推荐。也是本文主要讲述的内容。

   其中，前三者是和机器学习没有任何关系的，但却是推荐效果最好的三种方式。一般说来，这部分内容应该占到总的推荐内容的 80%左右，另外 20%则是对长尾内容的个性化推荐。

基于用户的协同过滤

如图 1 所示，在推荐系统中，用 m×n 的打分矩阵表示用户对物品的喜好情况，一般用打分来表示用户对商品的喜好程度，分数越高表示该用户对这个商品越感兴趣，而数值为空表示不了解或是没有买过这个商品。

商品 / 用户	电脑	连衣裙	U盘	棒棒糖
John	5		1	2
Tommy	3		4	

图 1 用于个性化推荐系统的打分矩阵

如图 2 所示，基于用户的协同过滤推荐算法是指找到与待推荐商品的用户 u 兴趣爱好最为相似的 K 个用户，根据他们的兴趣爱好将他们喜欢的商品视为用户 u 可能会感兴趣的商品对用户 u 进行推荐。

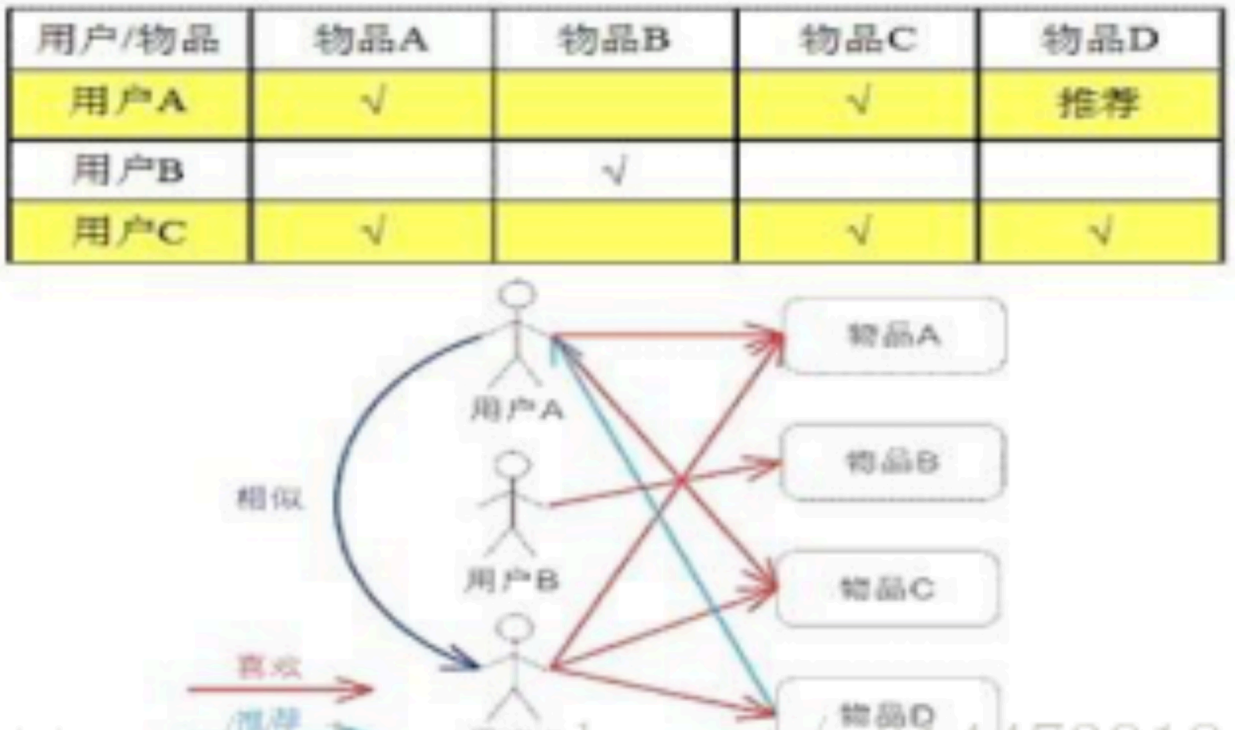


图 2 基于用户的协同过滤算法

基于用户的协同过滤推荐算法主要分为两步，

1. 第一步是求出用户之间的相似度

2. 第二步是根据用户之间的相似度找出与待推荐的用户最为相似的几个用户并根据他们的兴趣爱好向待推荐用户推荐其可能会感兴趣的商品。

用户之间的相似度的计算主要可以通过 Jaccard 公式和余弦相似度公式得到。

Jaccard（杰卡德相似性系数）公式如下：

$$w_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

余弦相似度公式为：

$$w_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)| \bullet |N(v)|}}$$

其中，N(u)为用户 u 感兴趣的商品，N(v)为用户 v 感兴趣的商品。

而计算用户 u 对商品 i 的感兴趣程度打分公式如下：

$$p(u,i) = \sum_{v \in S(u,K) \cap N(i)} w_{uv} r_{vi}$$

*S(u,K)*包含了和用户 u 兴趣最接近的 K 个用户，*N(i)*表示对商品 i 有过打分行为的用户集合，*Wuv* 表示计算出的用户 u 和用户 v 的兴趣相似度，*rvi* 表示用户 v 对商品 i 打的分数。

得到了用户 u 对所有未打分商品的感兴趣程度分数后，将分数最高的几个商品为用户 u 最有可能感兴趣的商品推荐给用户 u。

基于商品的协同过滤

基于用户的协同过滤基本思想非常简单，就是找到志同道合的朋友，并把朋友感兴趣的而用户没有接触过的商品推荐给用户。

但是这有一个问题，由于新用户的注册量非常高，基于用户的协同过滤推荐需要计算新用户和之前的用户之间的相似度，这会将数据稀疏，延展性差等问题暴露的非常明显。

所以基于商品的协同过滤方法被提出，相较于用户之间的相似度，商品之间的相似度相对是静态的，当新用户注册并有了一些自己感兴趣的商品信息时，无需再进行计算，直接根据之前存储的商品之间的相似度，将用户可能感兴趣的商品推荐给用户。

用户/物品	物品A	物品B	物品C
用户A	√		√
用户B	√	√	√
用户C	√		推荐

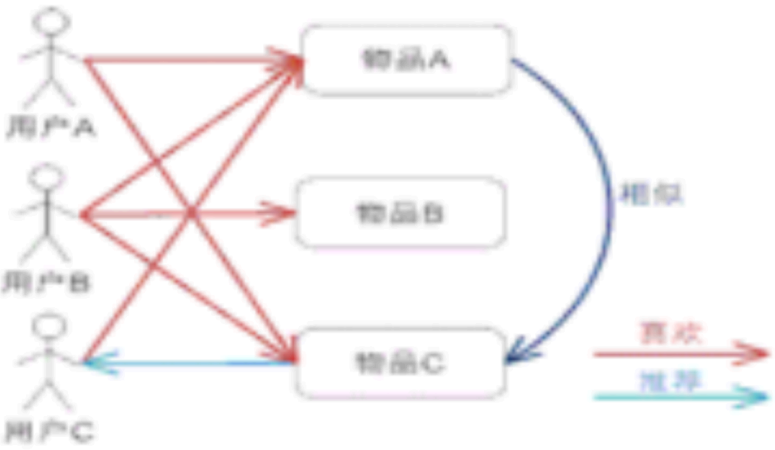


图 3 基于商品的协同过滤

基于商品的协同过滤推荐算法也是分为两步，

1. 第一步是根据数据库中已有的信息求出商品的相似度，
2. 第二步是利用求出的商品之间的相似度计算用户对某种商品可能的兴趣程度。

商品之间的相似度可以利用皮尔逊相似度，余弦相似度或是改进的余弦相似度来进行计算。

皮尔逊相似度：

$$sim(i,j)=\cos(\vec{i},\vec{j})=\frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2*\|\vec{j}\|_2}$$

U 表示所有用户的集合，*Ru,i* 表示用户 u 对商品 i 的打分，*Ru,j* 表示用户 u 对商品 j 的打分，*Ri* 和 *Rj* 分别表示用户对商品 i 和商品 j 的评分的平均值。

余弦相似度计算商品 i 和商品 j 的相似度的公式为：

$$sim(i,j)=\frac{\sum_{u\in U}(R_{u,i}-\overline{R}_u)(R_{u,j}-\overline{R}_u)}{\sqrt{\sum_{u\in U}(R_{u,i}-\overline{R}_u)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\overline{R}_u)^2}}$$

i 其中表示在打分矩阵中所有用户对商品 i 的打分（若没有打分则默认置为 0）构成的向量，j 表示打分矩阵中所有用户对商品 j 打分构成的向量。分子是两个向量的内积。

由于余弦相似度并没有考虑到用户打分尺度的问题，比如有人喜欢给高分而有人喜欢给低分，由于这种打分习惯的不同，有可能计算出来的相似度和实际差距很大，所以为了解决这个问题，提出了修正的余弦相似度计算。具体公式如下：

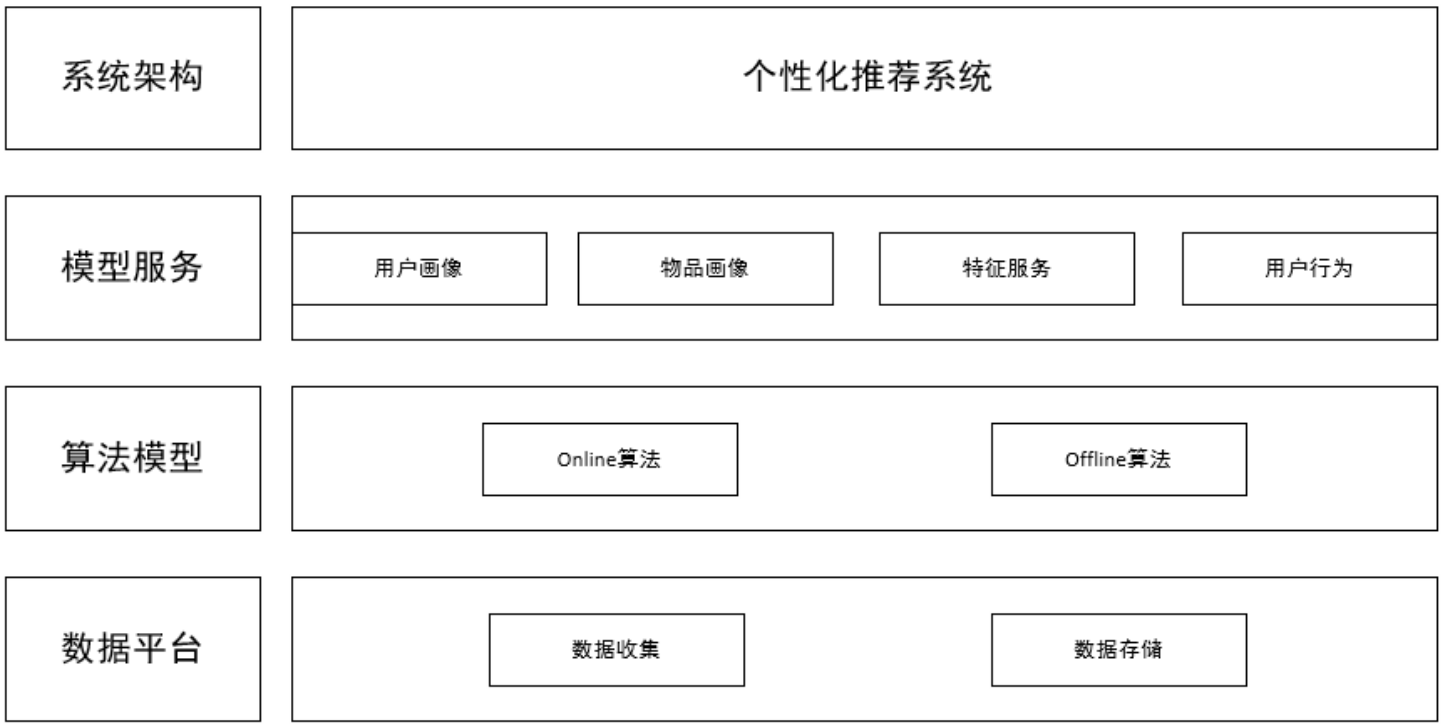
其中 *Ru,i* 和 *Ru,j* 表示用户 u 对商品 i 和 j 的打分，*Ru* 表示用户 u 对所有商品打分的平均值。

用于预测的打分函数公式则可以如下定义：

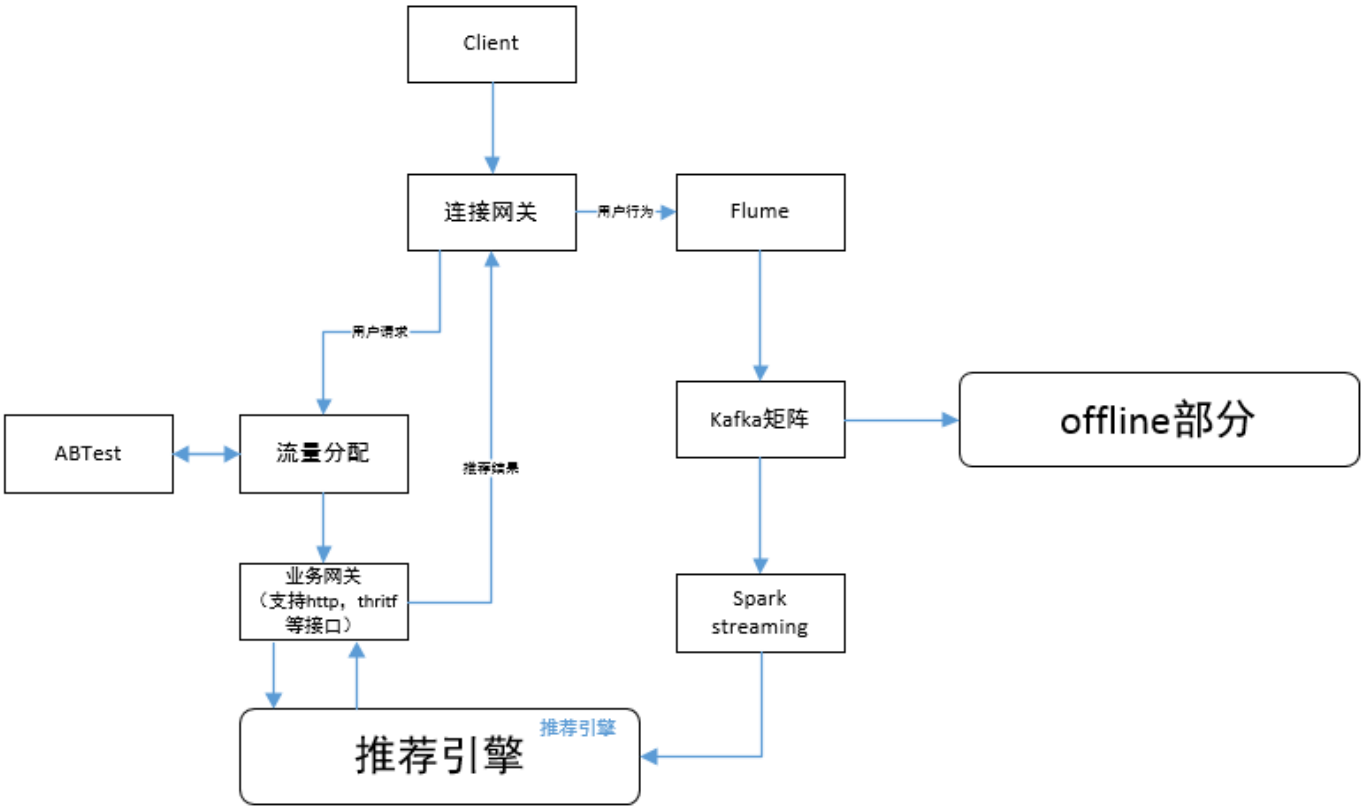
$$P_{u,i} = \frac{\sum_{\text{all similar items, N}}(s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}}(|s_{i,N}|)}$$

其中，*Si,N* 为物品 i 和物品 N 之间的相似度，*Ru,N* 为用户 u 对物品 N 的打分。这个公式的意义相当于是对用户 u 对商品 i 的可能得分取了一个加权平均。

推荐系统架构



online 部分架构



核心模块

业务网关，推荐服务的入口，负责推荐请求的合法性检查，组装请求响应的结果。

推荐引擎，推荐系统核心，包括 online 逻辑，召回、过滤、特征计算、排序、 多样化等处理过程。

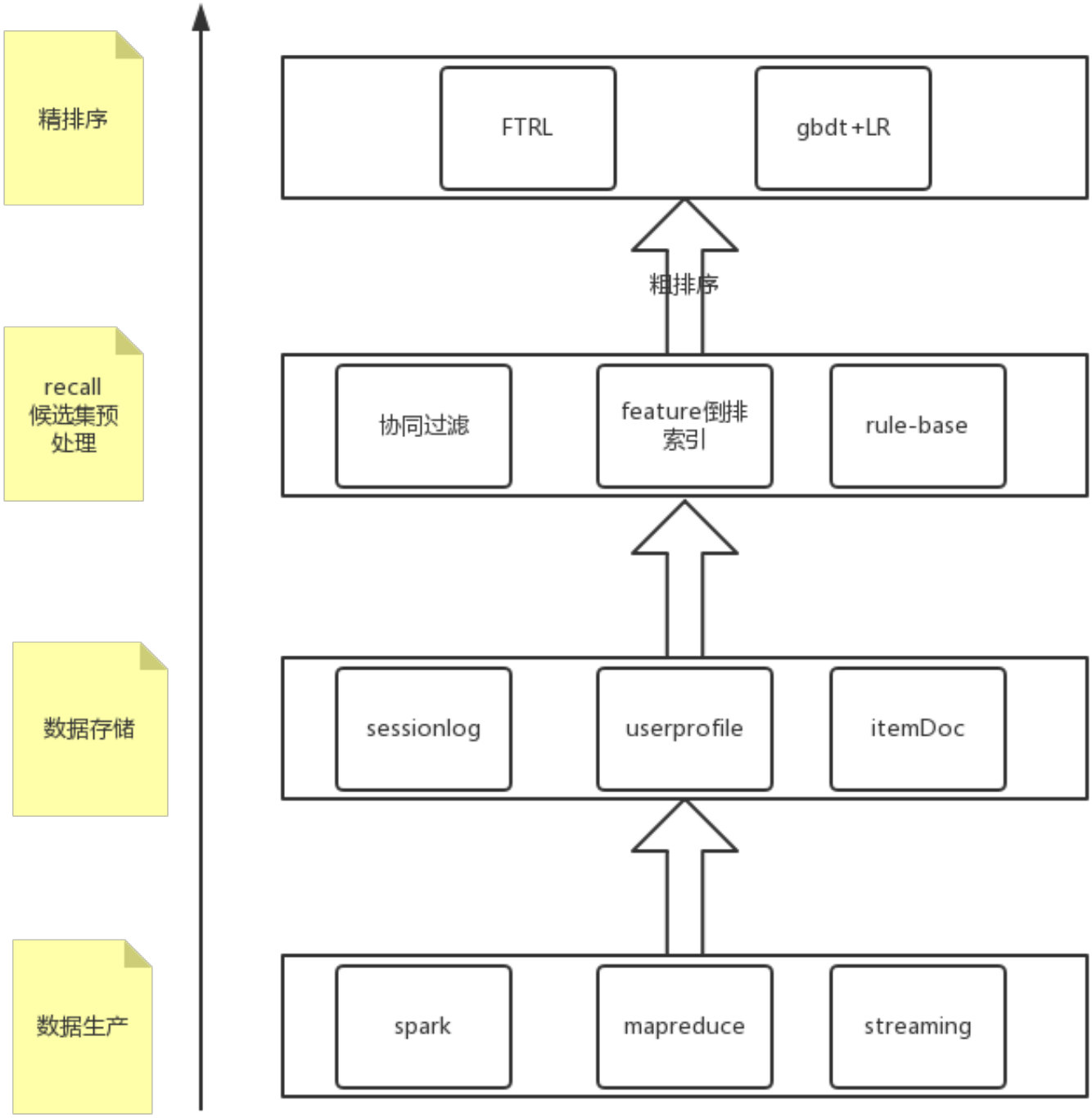
数据路径

1、请求的刷新从 gateway，经过流量分配模块，传到业务 gateway，业务 gateway 支持 http，tcp（使用 thirtf 协议或者 protobuf 协议）等多种类型接口；

2、用户行为数据，从 gateway 到 Flume agent，然后到 kafka，为后面 online，realtime userprofile 部分的提供实时数据，也为 offline 部分的数据存储系统提供数据。

offline 部分架构

一个推荐系统的主要部分



从框架的角度看，推荐系统基本可以分为数据层、召回层、排序层。

数据层包括数据生成和数据存储，主要是利用各种数据处理工具对原始日志进行清洗，处理成格式化的数据，落地到不同类型的存储系统中，供下游的算法和模型使用。

sessionlog：对原始数据进行清洗合并，sessionlog 一般就是清洗合并后的数据，后续的算法和统计都是根据 sessionlog 进行再加工。

userprofile：对用户属性和行为等信息进行采集和统计，为后续算法提供特征支持。

itemDoc：对视频、商品等属性、曝光、点击等字段进行统计， 为后续算法提供特征支持。

召回层主要是从用户的历史行为、实时行为等角度利用各种触发策略产生推荐的候选集，对不同的策略和算法产生的候选集进行融合并按照产品规则进行过滤，一般融合和过滤后的候选集还是比较多的，一次线上请求过来之后线上系统无法对那么多的候选集进行排序，所以在召回层一般还会有粗排序，对融合的候选集进行一次粗排序，过滤掉粗排分数较低的候选集。

排序层主要是利用机器学习的模型对召回层筛选出来的候选集进行精排序。

数据特征

数据决定了特征，特征决定了效果的上限，模型决定了接近效果上限的程度。

行为类别	**行为表现**
用户主动行为	点击、分享、评分
用户画像	用户属性（性别、年龄、收入）、视频分类兴趣分布、地域、时间
负反馈	负评

1. 用户主动行为数据记录了用户在平台的的各种行为，这些行为一方面用于候选集触发算法中的离线计算（主要是浏览、下单），另外一方面，这些行为代表的意图的强弱不同，因此在训练重排序模型时可以针对不同的行为设定不同的回归目标值，以更细地刻画用

户的行为强弱程度。此外，用户对 deal 的这些行为还可以作为重排序模型的交叉特征，用于模型的离线训练和在线预测。

2. 负反馈数据反映了当前的结果可能在某些方面不能满足用户的需求，因此在后续的候选集触发过程中需要考虑对特定的因素进行过滤或者降权，降低负面因素再次出现的几率，提高用户体验；同时在重排序的模型训练中，负反馈数据可以作为不可多得的负例参与模型训练，这些负例要比那些展示后未点击、未下单的样本显著的多。
3. 用户画像是刻画用户属性的基础数据，其中有些是直接获取的原始数据，有些是经过挖掘的二次加工数据，比如用户的聚类和向量化，这些属性一方面可以用于候选集触发过程中对 deal 进行加权或降权，另外一方面可以作为重排序模型中的用户维度特征。、

基于内容的召回

主要是以之前 NLP 得到的内容画像为基础，以 item 对应分类/主题/关键词的权重建立召回，依据用户画像的相应权重和内容画像的距离排序召回。

基于用户群

首先我们需要对用户分群，聚类的方案有很多，

1. 对 item 进行向量化（w2v）然后对 item 进行聚类，用户对 item 的行为就可以把 item 的簇赋值到 user 身上。
2. 直接对用户进行向量化，比如降维。

总之最终的目的就是将用户 embedding 成一个向量，然后在对用户向量进行聚类，一般 k-means 就可以胜任大部分的场景。

1. 倒排链

tag-itemList，对每个用户的 tag 进行遍历，然后通过倒排链快速找到含有该 tag 的 itemList 然后 topN 抽取。