从今天起，我们不再单独介绍推荐算法的原理，而是开始进入一个新的模块——工程篇。

在工程实践的部分中，我首先介绍的内容是当今最热门的信息流架构。

信息流是推荐系统应用中的当红炸子鸡，它表现形式有很多：社交网络的动态信息流、新闻阅读的图文信息流、短视频信息流等等。

如果要搭建一个自己的信息流系统，它应该是怎么样的呢？今天，我就来带你一探信息流架构的究竟。

整体框架

信息流，通常也叫作 feed，这个英文词也很有意思，就是“喂”给用户的意思。

传统的信息流产品知识简单按照时间排序，而被推荐系统接管后的信息流逐渐成为主流，按照兴趣排序，也叫作“兴趣 feed”。

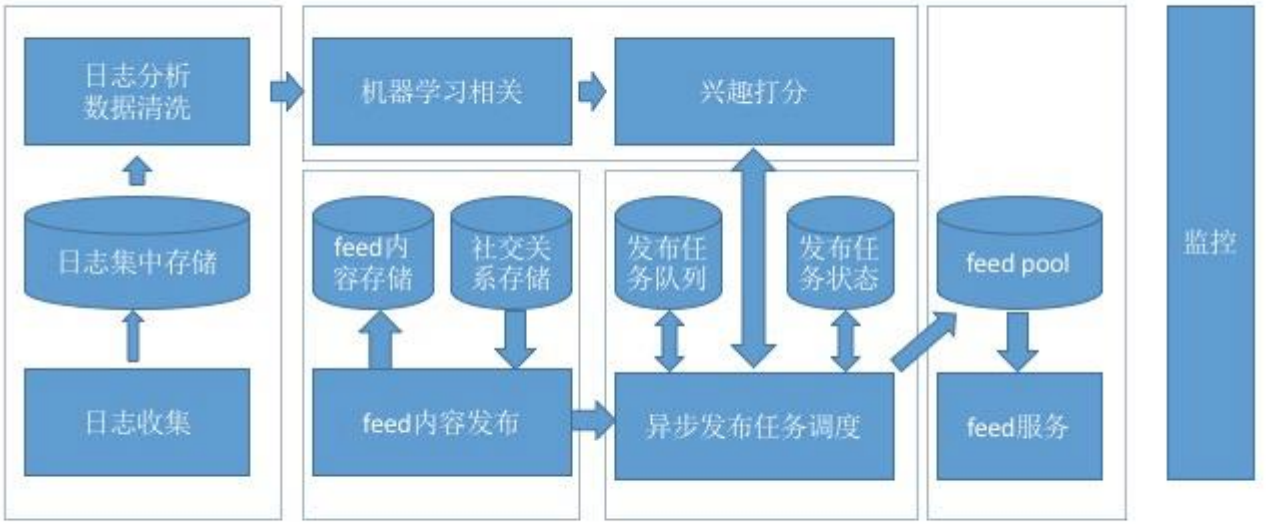
所以我们通常提到信息流，或者兴趣 feed，其实都是在说同一个话题。

这里温馨提示一下：如果要搜索 feed 相关的技术文章，你应该用“Activity Stream”作为关键词去搜，而不应该只用“feed”搜索，Activity Stream 之于 feed，就好比多潘立酮之于吗丁啉，前者是行话，后者是通俗说法。

要实现一个信息流，整体逻辑上是比较清楚的。可以划分为两个子问题。

1. 如何实现一个按照时间顺序排序的信息流系统？
2. 如何给信息流内容按照兴趣重排序？

我这里先给出一个整体的框架，然后再分别详谈。



这张架构图划分成几个大的模块：日志收集、内容发布、机器学习、信息流服务、监控。这里分别介绍一下：

1. 日志收集，是所有排序训练的数据来源，要收集的最核心数据就是用户在信息流上产生的行为，用于机器学习更新排序模型；
2. 内容发布，就是用推或者拉的模式把信息流的内容从源头发布到受众端；
3. 机器学习，从收集的用户行为日志中训练模型，然后为每一个用户即将收到的信息流内容提供打分服务；
4. 信息流服务，为信息流的展示前端提供 Rest API；
5. 监控，这是系统的运维标配，保证系统的安全和稳定等。

数据模型

信息流的基本数据有三个：用户（User）、内容（Activity）和关系（Connection）。用户不用说，就是区别不同用户的身份 ID，我来说一说其他的两种。

1. 内容即 Activity。

用于表达 Activity 的元素有相应的规范，叫作 Atom，你可以参考它并结合产品需求，定义出自己的信息流数据模型来。

根据 Atom 规范的定义，一条 Activity 包含的元素有：Time、Actor、Verb、Object、Target、Title、Summary。下面详细解释一下这些元素。

1. Time：即“Activity 发生的时间”。
2. Actor：即“Activity 由谁发出的”。通常 Actor 就是用户 ID，但是我们也可以扩展到其他拟人化物体上，如关注的一个“店铺”，收藏的一部“电影”，或者用户喜欢的一个标签或者分类。也就是和用户建立连接的另一端。
3. Verb：动词，就是连接的名字，比如“Follow”“Like”等，也可以是隐含的连接，如挖掘出的用户兴趣词和用户之间这种潜规则。
4. Object：即动作作用到最主要的对象，只能有一个，比如一个人赞过的一张照片，店铺上新的一件商品，一个分类下一篇新的文章。
5. Target：动作的最终目标，与 verb 有关，可以没有。它对应英语中介词 to 后接的事物，比如“John saved a movie to his wishlist”（John 保存了一部电影到清单里），这里电影就是 Object，而清单就是 Target。
6. Title：这个是 Activity 的标题，用自然语言描述，用于展示给用户。
7. Summary：通常是一小段 HTML 代码，是对这个 Activity 的描述，还可能包含类似缩略图这样的可视化元素，可以理解为 Activity 的视图，不是必须的。

举个例子：2016 年 5 月 6 日 23:51:01（Time）@刑无刀（Actor） 分享了（Verb）一条微博（Object） 给 @极客时间 （Target）。把前面这句话去掉括号后的内容就是它的 Title，Summary 暂略。

除了上面的字段外，还有一个隐藏的 ID，用于唯一标识一个 Activity。社交电商 Etsy 在介绍他们的信息流系统时，还创造性地给 Activity 增加了 Owner 属性，同一个 Activity 可以属于不同的用户，相当于考虑了方向。

2. 关系即连接。

互联网产品里处处皆连接，有强有弱，好友关系、关注关系等社交是较强的连接，还有点赞、收藏、评论、浏览，这些动作都可以认为是用户和另一个对象之间建立了连接。有了连接，就有信息流的传递和发布。

定义一个连接的元素有下面几种。

1. From：连接的发起方。
2. To：被连接方。
3. Type/Name：就是 Atom 模型中的 Verb，即连接的类型：关注、加好友、点赞、浏览、评论，等等。
4. Affinity：连接的强弱。

如果把建立一个连接视为一个 Activity 模型的话，From 就对应 Activity 中的 Actor，To 就对应 Activity 中的 Object。

连接的发起从 From 到 To，内容的流动从 To 到 From。Connection 和 Activity 是相互加强的，这是蛋和鸡的关系：有了 Activity，就会产生 Connection，有了 Connection，就可以“喂”（feed）给你更多的 Activity。

在数据存储上可以选择的工具有下面的几种：

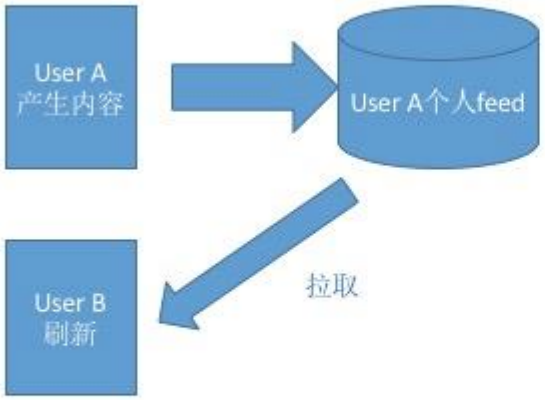
Activity 存储可以采用 MySQL、Redis、Cassandra 等； Connection 存储可以采用 MySQL； User 存储可以采用 MySQL。

内容发布

用户登录或者刷新后，信息流是怎么产生的呢？我们把内容出现在受众的信息流中这个过程称为 Fan-out，直觉上是这样实现的：

1. 获取用户所有连接的终点（如好友、关注对象、兴趣标签）；
2. 获取这些连接终点（关注对象）产生的新内容（Activity）；
3. 按照某个指标排序后输出。

上面这个步骤别看简单，在一个小型的社交网络上，通常很有效，而且 Twitter 早期也是这么做的。这就是江湖行话说的“拉”模式（Fan-out-on-load），即：信息流是在用户登录或者刷新后实时产生的。这里有一个示意图，你可以点击查看。



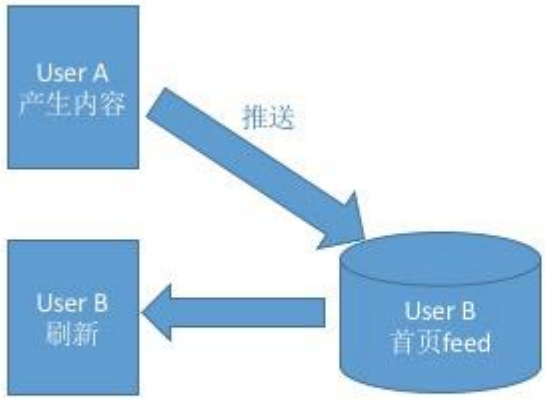
拉模式就是当用户访问时，信息流服务才会去相应的发布源拉取内容到自己的 feed 区来，这是一个阻塞同步的过程。“拉”模式的好处也显而易见，主要有下面两种。

1. 实现简单直接：一行 SQL 语句就搞定了。
2. 实时：内容产生了，受众只要刷新就看得见。

但是也有很大的不足：

1. 随着连接数的增加，这个操作的复杂度指数级增加；
2. 内存中要保留每个人产生的内容；
3. 服务很难做到高可用。

与“拉”模式对应，还有一个“推”模式（Fan-out-on-write）。我在文稿里放了一张图，你可以点击查看。



当一个 Actor 产生了一条 Activity 后，不管受众在不在线，刷没刷新，都会立即将这条内容推送给相应的用户（即和这个 Actor 建立了连接的人），系统为每一个用户单独开辟一个信息流存储区域，用于接收推送的内容。如此一来，当用户登录后，系统只需要读取他自己的信息流即可。

“推”模式的好处显而易见：在用户访问自己的信息流时，几乎没有任何复杂的查询操作，所以服务可用性较高。

“推”模式也有一些不足。

1. 大量的写操作：每一个粉丝都要写一次。
2. 大量的冗余存储：每一条内容都要存储 N 份（受众数量）。
3. 非实时：一条内容产生后，有一定的延迟才会到达受众信息流中。
4. 无法解决新用户的信息流产生问题。

既然两者各有优劣，那么实际上就应该将两者结合起来，一种简单的结合方案是全局的：

1. 对于活跃度高的用户，使用推模式，每次他们刷新时不用等待太久，而且内容页相对多一些；
2. 对于活跃度没有那么高的用户，使用拉模式，当他们登录时才拉取最新的内容；
3. 对于热门的内容生产者，缓存其最新的 N 条内容，用于不同场景下的拉取。

还有一种结合方案是分用户的，这是 Etsy 的设计方案：

1. 如果受众用户与内容产生用户之间的亲密度高，则优先推送，因为更可能被这个受众所感兴趣；
2. 如果受众用户与内容产生用户之间的亲密度低，则推迟推送或者不推送；
3. 也不是完全遵循亲密度顺序，而是采用与之相关的概率。

在中小型的社交网络上，采用纯推模式就够用了，结合的方案可以等业务发展到一定规模后再考虑。

对于信息流的产生和存储可以选择的工具有：

- 用户信息流的存储可以采用 Redis 等 KV 数据库，使用 uid 作为 key。
- 信息流推送的任务队列可以采用 Celery 等成熟框架。

信息流排序

信息流的排序，要避免陷入两个误区：

1. 没有目标；
2. 人工量化。

第一个误区“没有目标”意思就是说，设计排序算法之前，一定要先弄清楚为什么要对时间序重排？希望达到什么目标？只有先确定目标，才能检验和优化算法。

第二个误区是“人工量化”，也就是我们通常见到的产品同学或者运营同学要求对某个因素加权、降权。这样做很不明智，主要是不能很好地持续优化。

目前信息流采用机器学习排序，以提升类似互动率，停留时长等指标，这已经成为共识。比如说提高互动率则需要下面几个内容。

•

首先，定义好互动行为包括哪些，比如点赞、转发、评论、查看详情等；

•
•

其次，区分好正向互动和负向互动，比如隐藏某条内容、点击不感兴趣等是负向的互动。

•

基本上到这里就可以设计成一个典型的二分类监督学习问题了，对一条信息流的内容，在展示给用户之前，预测其获得用户正向互动的概率，概率就可以作为兴趣排序分数输出。

能产生概率输出的二分类算法都可以用在这里，比如贝叶斯、最大熵、逻辑回归等。

互联网常用的是逻辑回归（Logistic Regression），谁用谁知道，用过的都说好；也有 Facebook 等大厂采用了逻辑回归加梯度提升树模型（又称 GBDT）来对信息流排序，效果显著。

如今大厂都已经转向深度学习了，但我还是建议小厂或者刚起步的信息流先采用线性模型。

对于线性模型，一个重要的工作就是特征工程。信息流的特征有三类：

1. 用户特征，包括用户人口统计学属性、用户兴趣标签、活跃程度等；
2. 内容特征，一条内容本身可以根据其属性提取文本、图像、音频等特征，并且可以利用主题模型提取更抽象的特征。
3. 其他特征，比如刷新时间、所处页面等。

排序模型在实际使用时，通常做成 RPC 服务，以供发布信息流时调用。

数据管道

信息流是一个数据驱动的系统，既要通过历史数据来寻找算法的最优参数，又要通过新的数据验证排序效果，所以搭建一个数据流管道就是大家翘首期盼的。

这个管道中要使用的相关数据可能有：

1. 互动行为数据，用于记录每一个用户在信息流上的反馈行为；
2. 曝光内容，每一条曝光要有唯一的 ID，曝光的内容仅记录 ID 即可；
3. 互动行为与曝光的映射关系，每条互动数据要对应到一条曝光数据；
4. 用户画像内容，即用户画像，提供用户特征，具体请见我在第 4、5、6 三篇中的内容；
5. 信息流的内容分析数据，提供内容特征，即物品画像。

对于一个从零开始的信息流，没必要做到在线实时更新排序算法的参数，所以数据的管道可以分成三块：

1. 生成训练样本，可离线；
2. 排序模型训练，可离线；
3. 模型服务化，实时服务；

像 Pinterest 早期的管道也差不多就是这样。

在离线训练优化模型时，关注模型的 AUC 是否有提升，线上 AB 测试时关注具体的产品目标是否有提升，比如互动率等，同时还要根据产品具体形态关注一些辅助指标。

另外，互动数据相比全部曝光数据，数量会小得多，所以在生成训练数据时需要对负样本（展示了却没有产生互动的样本）进行采样，采样比例也是一个可以优化的参数。

固定算法和特征后，在 0.1~0.9 之间遍历对比实验，选择最佳的正负比例即可。经验比例在 2:3 左右，即负样本略大于正样本，你可以用这个比例做启发式搜索。

总结

今天我逐一梳理了实现一个通用信息流的关键模块，及其已有的轮子，从而能最大限度地降低开发成本。

这些对于一个中小型的社交网络来说已经足够，当你面临更大的社交网络，会有更多复杂的情况出现，尤其是系统上的。

所以，壮士，请好自为之，时刻观察系统的监控、日志的规模。

你在了解了典型的信息流架构之后，可以说一说 Facebook 这样的社交网络 feed，和头条这样的资讯信息流之间的差别和共同点吗？欢迎给我留言。