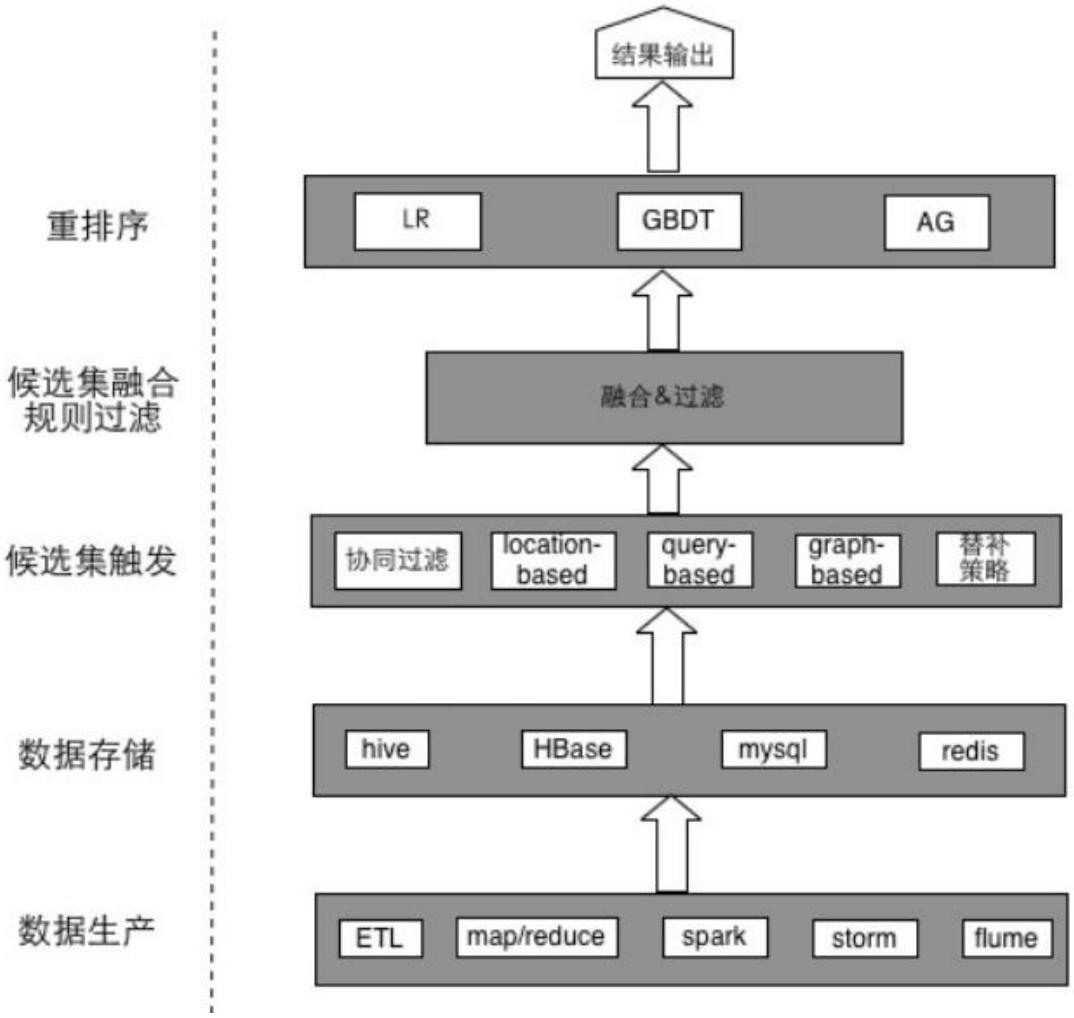
1.框架



从框架的角度看，推荐系统基本可以分**为数据层、触发层、融合过滤层和排序层**。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**。候选集触发层主要是从用户的历史行为、实时行为、地理位置等角度利用各种触发策略产生推荐的候选集。候选集融合和过滤层有两个功能，一是对出发层产生的不同候选集进行融合，提高推荐策略的覆盖度和精度；另外还要承担一定的过滤职责，**从产品、运营的角度确定一些人工规则**，过滤掉不符合条件的 item。**排序层**主要是利用机器学习的模型**对触发层筛选出来**的候选集进行重排序。

首先将客户上报过来的数据进行**数据清洗**，检查数据的一致性，处理无效值

和缺失值等，去除脏数据，处理成格式化数据存储到不同类型的存储系统中。对于用户行为日志和推荐日志由于随时间积累会越来越大，一般存储在分布式文件系统(HDFS)，即 Hive 表中，当需要的时候可以下载到本地进行离线分析。对于物品信息一般存储在 MySQL 中，但是对于业务数据，越来越多的客户导致物品信息表(item_info)越来越大，所以同时也会保存在 Hive 表和 HBase 中，Hive 可以方便离线分析时操作，但实时程序读取的时候 Hive 表的实时性较差，所以同时也会写一份放在 HBase 中供实时程序读取。对于各个程序模块生成的结果，有进程同步关系的程序一般会使用 Redis 作为缓冲存储，生产者会把信息写到 redis 中供消费者使用。**候选集生**成是从用户的历史行为、实时行为、利用各种策略和算法生成推荐的候选集。同时**点击反馈**会根据用户的实时操作对候选集进行实时的调整，对于部分新用户和历史行为不太丰富的用户，由于候选集太小，需要一些替补策略进行补充。候选集融合规则过滤主要有两个功能，一是对生成的候选集进行融合，提高推荐策略的覆盖度和精度;另外还需根据产品、运营的角度确定一些人为的规则，过滤掉不符合条件的 item，重排序主要是利用机器学习的模型对融合后的候选集进行重排序。

同时，对与**候选集触发和重排序两**层而言，为了效果迭代是需要频繁修改的两层，因此需要支持 **ABtest**。为了支持高效率的迭代，我们对候选集触发和重排序两层进行了解耦，这两层的结果是正交的，因此可以分别进行对比试验，不会相互影响。**同时在每一层的内部，我们会根据用户将流量划分为多份，支持多个策略同时在线对比。**

2. 机器学习重排序

对于不同算法触发出来的候选集，如果只是根据算法的历史效果决定算法产生的 item 的位置显得有些简单粗暴，同时，在每个算法的内部，不同 item 的顺序也只是简单的由一个或者几个因素决定，**这些排序的方法只能用于第一步的初选过程，最终的排序结果**需要借助机器学习的方法，使用相关的**排序模型**，综合多方面的因素来确定。

排序模型分为**非线性模型和线性模型**，非线性模型能较好的捕捉特征中的非

线性关系，但训练和预测的代价相对线性模型要高一些，这也导致了**非线性模型的更新周期相对要长**。相较而言，**线性模型对特征的处理要求比较高(LR 对特征要求较高)**，需要凭借领域知识和经验人工对特征做一些先期处理，但因为线性模型简单，在训练和预测时效率较高。因此在更新周期上也可以做的更短，还可以结合业务做一些在线学习的尝试。

2.1 线性模型

线性模型主要介绍**逻辑回归(Logistic Regression)**，逻辑回归是一种**广义线性模型**，虽然名字里带着回归，但它其实是一种分类算法，主要运用在二分类或多分类算法。在多分类中，有 one-vs-rest(OvR)，和 many-vs-many(MvM)两种不同的分类思路，这里主要讨论预测而分类问题(某个 userid 是否会点击某个 itemid)。**首先将每个 userid 和每个 itemid 作为特征**，模型函数为：

g(Z)=w0+w1*x1+w2*x2　　h(Z)=1/(1+e-g(Z))

m,k 分别为 userid 和 itemid 的个数，αi，βj 分别为自变量 Ui，Ij 的参数。逻辑回归模型采用**极大似然法**对模型的参数进行估计，Cost function 为：
交叉熵损失

n 为样本个数，yi 为样本的 label，用 θ 向量代替所有参数。然后是计算 Cost function 最大化时的参数。在最优化理论中，求解最优化参数有很多种方法，**梯度下降、随机梯度下降、牛顿法、拟牛顿法，共轭梯度法**，这里选用的是牛顿法。

2.2 非线性模型

非线性模型主要介绍 GBDT(Gradient Boost Decision Tree)，以及相应的运用。GBDT 是一种常用的非线性模型，是 Boost 算法的一种，先了解 AdaBoost 的最流行的元算法。

Adaboost 算法在开始的时候先为每个样本赋一个权重值，初始的时候，每个样本权重相同。每次迭代建立一个单层决策树分类器(可以用任意分类器作为弱分类器，只要它比随机猜测略好就行，不过弱分类器越简单越好)，该分类器依据计算预测样本的最小错误率选出最佳单层决策树，同时增加分错的点的权重，减少分对的点的权重，这样使得某些点如果老是被分错，那么就会被“严重关注”，

也就被赋上一个很高的权重。然后进行 N 次迭代(由用户指定)，将会得到 N 个简单的分类器(basic learner)，然后我们将它们组合起来(比如说可以对它们进行加权、或者让它们进行投票等)，得到一个最终的模型。

原始的 Boost 算法是在算法开始的时候，为每一个样本赋上一个权重值，初始的时候，大家都是一样重要的。在每一步训练中得到的模型，会使得数据点的估计有对有错，我们就在每一步结束后，增加分错的点的权重，减少分对的点的权重，这样使得某些点如果老是被分错，那么就会被“严重关注”，也就被赋上一个很高的权重。然后等进行了 N 次迭代(由用户指定)，将会得到 N 个简单的分类器(basic learner)，然后我们将它们组合起来(比如说可以对它们进行加权、或者让它们进行投票等)，得到一个最终的模型。

而 Gradient Boost 与传统的 Boost 的区别是，*每一次的计算是为了减少上一次的残差(residual)，而为了消除残差，我们可以在残差减少的梯度(Gradient)方向上建立一个新的模型。*所以说，在 Gradient Boost 中，每个新的模型的简化是为**了使得之前模型的残差往*梯度方向*减少**，与传统 Boost 对正确、错误的样本进行加权有着很大的区别。

具体的算法为：

$$\{\mathbf{x}_i, y_i\}^N = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1k} & y_1 \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2k} & y_2 \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3k} & y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \cdots & x_{Nk} & y_N \end{pmatrix}$$

我们的目标是在样本空间上，找到最优的预测函数 f*(X)，使得 X 映射到 y 的损失函数 L(y,F(X))达到最小，即：

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y,\mathbf{x}}\left[L(y, F(\mathbf{x}))\right]$$

损失函数的平方误差：

$$L(y, F(\mathbf{x})) = (y - F(\mathbf{x}))^2$$

假设预测函数 F(X)以 P={P1,P2,…} 为参数，并可以写成若干个弱分类器相加的形式，其中 P={βm, αm}0M,第 m 个弱分类器的表达形式为βmh(X;αm)，其中βmh(X;αm)

表示第 m 棵回归树，向量 α m 表示第 m 棵回归树的参数， β m 表示第 m 棵回归树在预测函数中的权重：

$$F(\mathbf{x};\mathbf{P})=\sum_{m=1}^{M}\beta_m h(\mathbf{x};\boldsymbol{\alpha}_m)$$

那么对于 N 个样本点{xi,yi}N，其优化问题等价于找到参数{ β m, α m}，m=0,1,2,…,M,使得：

$$(\beta_m,\boldsymbol{\alpha}_m)=\arg\min_{\boldsymbol{\alpha},\beta}\sum_{i=1}^{N}L(y_i,F_{m-1}(\mathbf{x}_i)+\beta h(\mathbf{x}_i;\boldsymbol{\alpha}))$$

求解归为以下迭代过程：

1. 首先定义初始化分类器为常数 ρ，其中 F0(X), 表示初始化弱分类器，常数 ρ，使得初始预测损失函数达到最小值：

$$F_0(\mathbf{x})=\arg\min_{\rho}\sum_{i=1}^{N}L(y_i,\rho)$$

2. 在每次迭代中都构造一个基于回归树的弱分类器，并设第 m 次迭代后得到的预测函数为 Fm(X), 相应的预测函数为 L(y, Fm(X))，为使预测损失函数减小得最快，第 m 个弱分类器 β mh(X; α m)应建立在前 m-1 次迭代生成的预测损失函数的梯度方向，其中-gm(xi)表示第 m 次迭代的弱分类器的建立方向，L(yi, F(xi))表示前 m-1 次迭代生成的预测损失函数，表达式为 L(yi, F(xi))=((yi-F(xi))2):

$$-g_m(\mathbf{x}_i)=-\left[\frac{\partial L(y_i,F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)},\quad i=1\cdots N$$

基于求得的梯度下降方向，参数 α m 是使回归树 h(X; α m)沿此方向逼近的参数值，即：

$$\boldsymbol{\alpha}_m=\arg\min_{\boldsymbol{\alpha},\beta}\sum_{i=1}^{N}[-g_m(\mathbf{x}_i)-\beta h(\mathbf{x}_i;\boldsymbol{\alpha})]^2$$

β m 是沿此方向搜索的最优步长，即：

3. 更新每次迭代后得到的预测函数，**即 Fm(X)= Fm-1(X)+ β mh(X; α m)**，若相应的预测损失函数满足误差收敛条件或生成的回归树达到预设值 M，则终止迭代。

4. 为避免过拟合现象，通常在每个弱分类器前乘上“学习速率”ν，值域为 0～1，ν 值越小，学习越保守，达到同样精度需要的迭代次数越大，反之，学习越

快速，越容易出现过拟合：

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \beta_m h(\mathbf{x}; \mathbf{a}_m)$$

2.3GBDT+LR

值得一提的是，GBDT 天然具有的优势是可以发现多种有区分性的特征以及特征组合。我们可以将 GBDT 和 LR 结合起来，具体如下：

先用已有特征训练 GBDT 模型，然后**利用 GBDT 模型学习到的树来构造新特征**，最后**把这些新特征加入原有特征一起训练模型**。**构造的新特征向量是取值 0/1 的，向量的每个元素对应于 GBDT 模型中树的叶子结点**。当一个样本点通过某棵树最终落在这棵树的一个叶子结点上，那么在新特征向量中这个叶子结点对应的元素值为 1，而这棵树的其他叶子结点对应的元素值为 0。新特征向量的长度等于 GBDT 模型里所有树包含的叶子结点数之和。

举例说明。下面的图中的两棵树是 GBDT 学习到的，第一棵树有 3 个叶子结点，而第二棵树有 2 个叶子节点。对于一个输入样本点 x，如果它在第一棵树最后落在其中的**第二个叶子**结点，而在第二棵树里最后落在**其中的第一个叶子结点**。**那么通过 GBDT 获得的新特征向量为[0, 1, 0, *1, 0*]**，其中向量中的前三位对应第一棵树的 3 个叶子结点，后两位对应第二棵树的 2 个叶子结点。

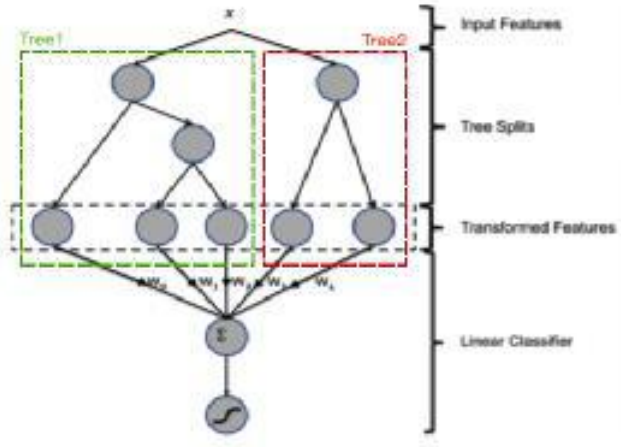


Figure 1: Hybrid model structure. Input features are transformed by means of boosted decision trees. The output of each individual tree is treated as a categorical input feature to a sparse linear classifier. Boosted decision trees prove to be very powerful feature transforms.

LR 虽然简单，且训练预测效率高，**但特征工程非常重要**，现有的特征工程实验，主要集中在寻找到有**区分度的特征、特征组合**，折腾一圈未必会带来效果提升。**GBDT 算法的特点正好可以用来发掘有区分度的特征、特征组合，减少特征工程中人力成本**。2014 Kaggle CTR 竞赛冠军就是使用这种组合方法，笔者也是向他们学习。

1. 排序模型

推荐机器学习排序算法演进

1、第一个时期

在一些互联网公司一般第一次上线机器学习排序模型时，选用了比较简单的 Logistic Regression，将重点放到架构设计上，尽量保证架构的正确性。除此之外，LR 模型的解释性强，方便 debug，并且通过特征权重可以解释推荐的内容，找到模型的不足之处。

在模型训练之前，我们首先解决的是评测指标和优化目标的问题。

评测指标（metrics)

线上效果的评测指标需要与长远目标相匹配，比如使用用户的投入程度和活跃度等。在我们的实验中，业界流行的 CTR 并不是一个好的评测指标，它会更偏向于较短的视频，标题党和低俗内容。

离线评测指标是按照业务来定制的，以便与在线评测指标匹配，这样在离线阶段就能够淘汰掉无效策略，避免浪费线上流量。

优化目标（objective）

机器学习会按照优化目标求解最优解，如果优化目标有偏差，得到的模型也存在偏差，并且在迭代中模型会不断地向这个偏差的方向学习，偏差会更加严重。**我们的方法是给样本添加权重，并且将样本权重加到 loss function 中**，使得优化目标与评测指标尽可能的一致，达到控制模型的目的。

LR 是个线性分类模型，要求输入是线性独立特征(比如 01 的度热编码的形式-对应 GBDT 处理后转换的数据)。我们使用的稠密的特征（维度在几十到几百之间）往往都是非线性的，并且具有依赖性，因此需要对特征进行转换。

特征转换需要对特征的分布，特征与 label 的关系进行分析，然后采用合适的转换方法。我们用到的有以下几种：**Polynomial Transformation(sklearn 中实现)**，Logarithmic or Exponential Transformation，Interaction Transformation 和 Cumulative Distribution Function 等。

虽然 LR 模型简单，解释性强，不过在特征逐渐增多的情况下，劣势也是显而易见的。

1、特征都需要人工进行转换为线性特征，十分消耗人力，并且质量不能保证

2、特征两两作 Interaction 的情况下，模型预测复杂度是平方项。在 100 维稠密特征的情况下，就会有组合出 10000 维的特征，复杂度高，增加特征困难。

3、三个以上的特征进行 Interaction 几乎是不可行的

2、第二个时期

为了解决 LR 存在的上述问题，我们把模型升级为 Facebook 的 GBDT+LR 模型，模型结构如图所示。

GBDT 是基于 Boosting 思想的 ensemble 模型，由多颗决策树组成，具有以下优点：

1、对输入特征的分布没有要求

2、根据熵增益自动进行特征转换、特征组合、特征选择和离散化，得到高维的组合特征，省去了人工转换的过程，并且支持了多个特征的 Interaction

3、预测复杂度与特征个数无关

假设特征个数 n=160 决策数个数 k=50，树的深度 d=6，两代模型的预测复杂度对比如下，升级之后模型复杂度降低到原来的 2.72%

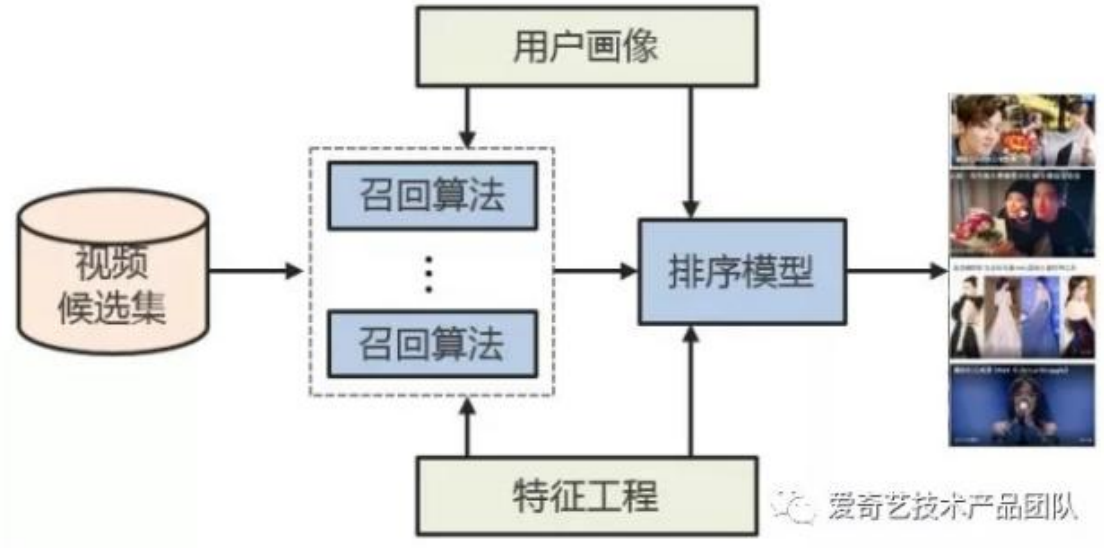
GBDT 与 LR 的 stacking 模型相对于只用 GBDT 会有略微的提升，更大的好处是防止 GBDT 过拟合。**升级为 GBDT+LR 后，线上效果提升了约 5%**，并且因为省去了对新特征进行人工转换的步骤，增加特征的迭代测试也更容易了。

2. 爱奇艺推荐

1、爱奇艺推荐系统介绍

我们的推荐系统主要分为两个阶段，召回阶段和排序阶段。

召回阶段根据用户的兴趣和历史行为，同千万级的视频库中挑选出一个小的候选集（几百到几千个视频）。这些候选都是用户感兴趣的内容，排序阶段在此基础上进行更精准的计算，能够给每一个视频进行较精确打分，进而从成千上万的候选中选出用户最感兴趣的少量高质量内容（十几个视频）。



推荐系统的整体结构如图所示，各个模块的作用如下：

1、用户画像：包含用户的人群属性、历史行为、兴趣内容和偏好倾向等多维度的分析，是个性化的基石

2、特征工程：包含了了视频的类别属性，内容分析，人群偏好和统计特征等全方位的描绘和度量，是视频内容和质量分析的基础

3、召回算法：包含了多个通道的召回模型，比如**协同过滤，主题模型，内容召回和 SNS 等通道**，能够从视频库中选出多样性的偏好内容

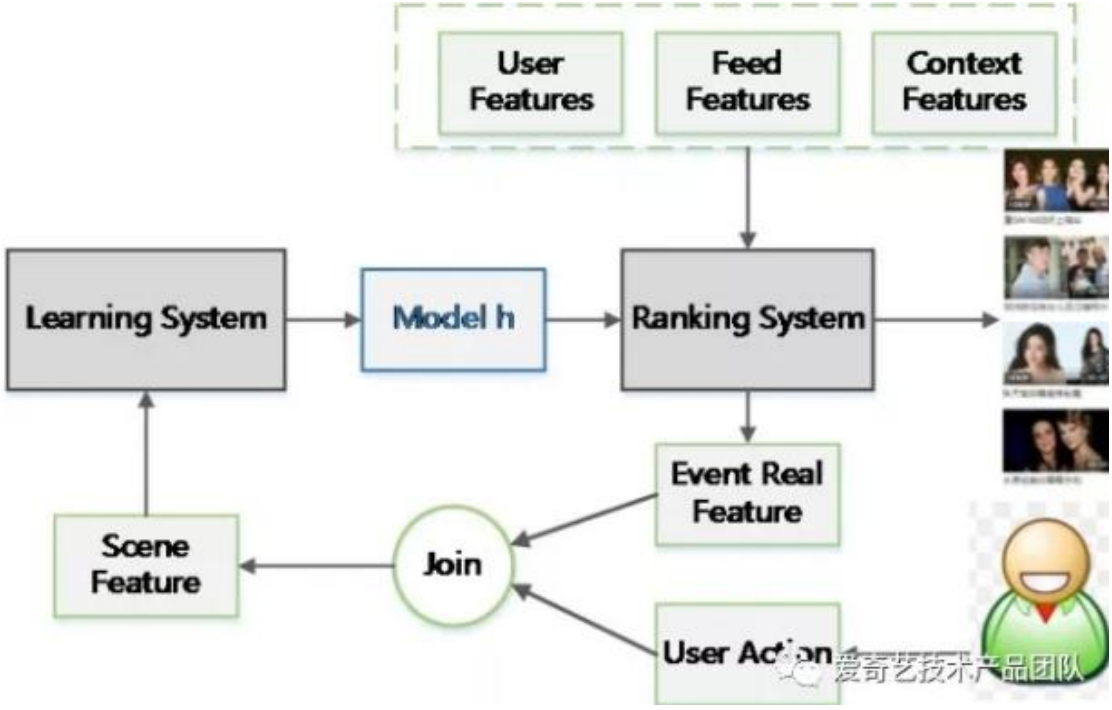
4、排序模型：对多个召回通道的**内容进行同一个打分排序**，选出最优的少量结果。

除了这些之外推荐系统还兼顾了推荐结果的多样性，新鲜度，逼格和惊喜度等多个维度，更能够满足用户多样性的需求。

2、推荐排序系统架构

在召回阶段，**多个通道的召回的内容是不具有可比性的，**并且因为数据量太大也难以进行更加较精确的偏好和质量评估，因此需要在排序阶段对召回结果进行统一的准确的打分排序。

(规则排序)用户对视频的满意度是有很多维度因子来决定的，这些因子在用户满意度中的重要性也各不相同，甚至各个因子之间还有多层依赖关系，人为制定复杂的规则既难以达到好的效果，又不具有可维护性，这就**需要借助机器学习的方法，使用机器学习模型来综合多方面的因子进行排序（基于模型排序效果）。**



排序系统的架构如图所示，主要由**用户行为收集，特征填充，训练样本筛选，模型训练，在线预测排序**等多个模块组成。

机器学习的主体流程是比较通用的，设计架构并不需要复杂的理论，更多的是需要对细节，数据流和架构逻辑的仔细推敲。

这个架构设计吸取了以前的经验和教训，在通用机器学习的架构基础上解决了两个问题：

训练预测的一致性

机器学习模型在训练和预测之间的差异会对模型的准确性产生很大的影响，尤其是模型训练与在线服务时特征不一致，比如用户对推荐结果的反馈会实时影响到用户的偏好特征，在训练的时候用户特征的状态已经发生了变化，模型如果依据这个时候的用户特征就会产生非常大的误差。

我们的解决办法是，将在线服务时的特征保存下来，然后填充到收集的用户行为样本中，这样就保证了训练和预测特征的一致性。

持续迭代

互联网产品持续迭代上线是常态，在架构设计的时候，数据准备，模型训练和在线服务都必须能够对持续迭代有良好的支持。

我们的解决方案是，数据准备和模型训练各阶段解耦，并且策略配置化，这种架构使模型测试变得非常简单，可以快速并行多个迭代测试。