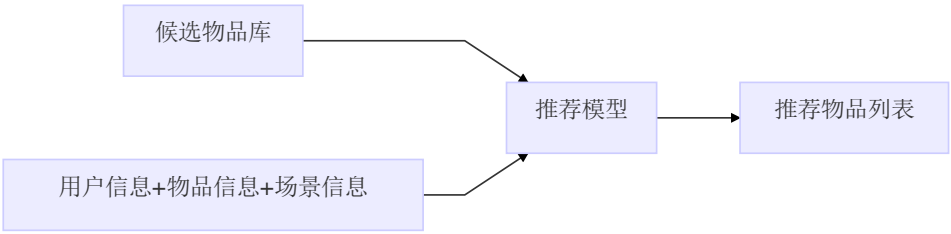
推荐系统的架构

互联网企业的核心需求是“增长”，推荐系统处于“增长引擎”的核心位置

推荐系统要解决的问题是用户如何在”信息过载“的情况下高效获得感兴趣的信息

1. **推荐系统的逻辑框架**



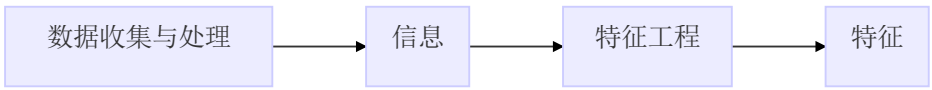
对于用户$U(user)$，在特定场景$C(context)$下，针对海量物品信息，构造函数$f(U,I,C)$，预测用户对特定物品$I(item)$的喜好程度，根据喜好程度对候选物品进行排序，进而生成推荐列表

2. **推荐系统的技术架构**

    1. **数据和信息**相关的问题。如何定义、存储、更新和处理用户信息、物品信息、场景信息？逐步发展为融合了数据离线批处理、实时流处理的数据流框架
    2. **算法和模型**相关的问题。推荐模型如何训练、如何预测、如何产生更好的效果？进一步细化为推荐系统中集训练、评估、部署、线上推断为一体的模型框架

3. **推荐系统的数据部分**

    主要负责“用户”“物品”“场景”的信息收集和处理

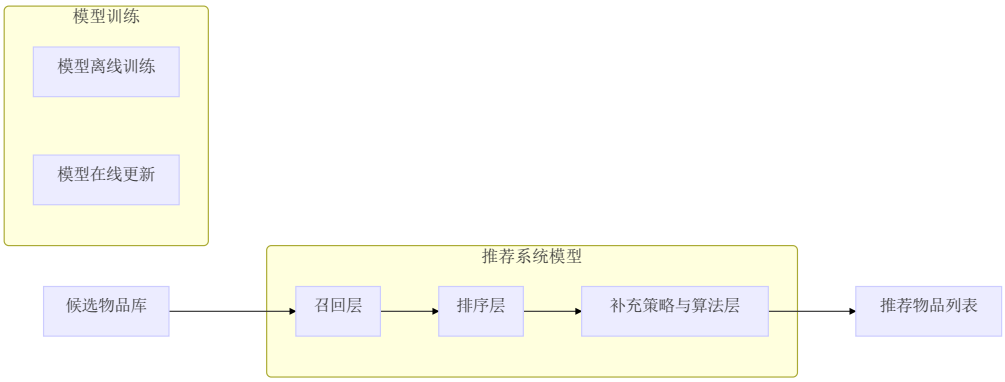


负责**数据收集与处理**的平台按照实时性由强到弱和数据处理能力由弱到强的顺序排序，依次为“客户端及服务器实时数据处理”“流处理平台准实时数据处理”“大数据平台离线数据处理”

得到原始数据后，数据处理系统会将原始数据进一步加工

4. **推荐系统的模型部分**

    是推荐系统的**主体**



召回层利用高效的召回规则、算法、模型，快速从海量候选物品中召回可能感兴趣的物品

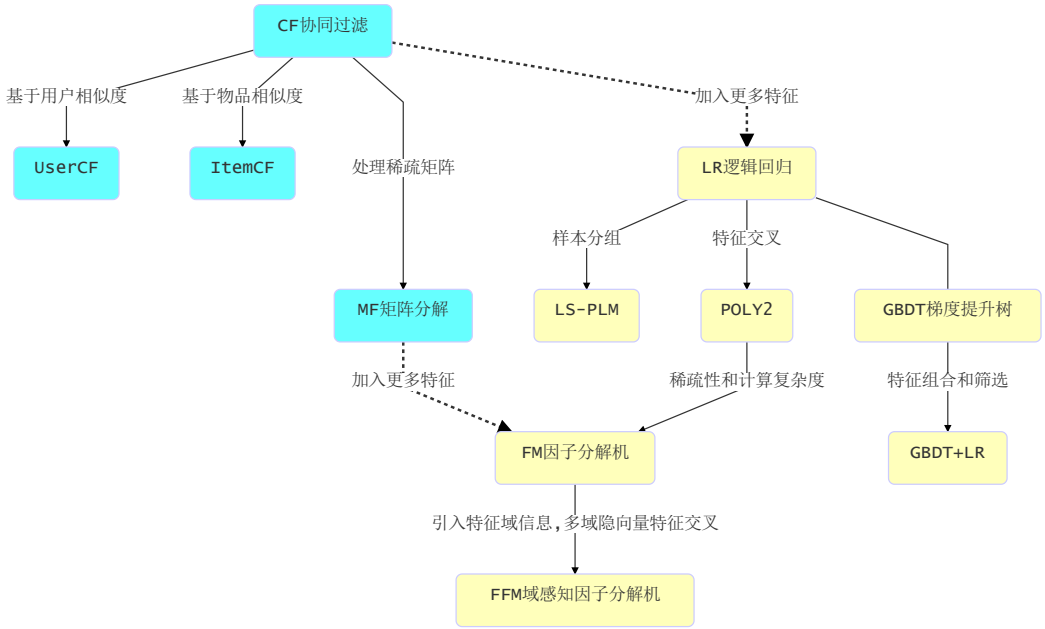
排序层利用排序模型对初筛的候选物品进行精排序（**重点**）

补充策略与算法层可以在将推荐列表返回用户之前，结合一些补充的算法对列表进行调整

模型训练用来确定模型结构以及相关算法和策略中的参数取值，离线训练可以利用全量样本使模型逼近全局最优点，在线更新可以准实时"消化"新的数据样本，更快反映新的数据结构变化趋势

5. **深度学习对推荐系统的贡献**
    - 在于对推荐模型部分的改进
    - 对数据模式的拟合能力更强，对特征组合的挖掘能力更强，具有更强的灵活性
    - 对推荐系统的数据流部分提出了新的挑战

传统推荐模型的演化关系图



协同过滤

1. **UserCF**

	物品1	物品2	物品3
用户A	赞	踩	赞
用户B		赞	踩
用户C	踩		赞
用户X	赞	?	踩

目的：预测用户X对物品2的评价，进而确定是否将物品2推荐给物品X

赋值：赞=1，踩=-1，未评价=0

用户相似度计算：

1. **余弦相似度**

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

2. **皮尔逊相关系数**

$$sim(i,j) = \frac{\sum_{p \in P}(R_{i,p} - \bar{R}_i)(R_{j,p} - \bar{R}_j)}{\sqrt{\sum_{p \in P}(R_{i,p} - \bar{R}_i)^2}\sqrt{\sum_{p \in P}(R_{j,p} - \bar{R}_j)^2}}$$

$R_{i,p}$代表用户i对物品p的评分，$\bar{R}_i$代表用户i对所有物品的平均评分，P代表所有物品的集合

减少了用户评分偏置的影响

3. 在皮尔逊相关系数的基础上引入**物品平均分**

$$sim(i,j) = \frac{\sum_{p \in P}(R_{i,p} - \bar{R}_p)(R_{j,p} - \bar{R}_p)}{\sqrt{\sum_{p \in P}(R_{i,p} - \bar{R}_p)^2}\sqrt{\sum_{p \in P}(R_{j,p} - \bar{R}_p)^2}}$$

$\bar{R}_p$代表物品p得到所有评分的平均分

减少了物品评分偏置的影响

最终结果的排序：

$$R_{u,p} = \frac{\sum_{s \in S}(w_{u,s} R_{s,p})}{\sum_{s \in S} w_{u,s}}$$

权重$w_{u,s}$是用户u和用户s的相似度，即$sim(u,s)$。$R_{s,p}$是用户s对物品p的评分

缺陷：

1. 用户相似度矩阵的存储空间是$S(n^2)$，互联网飞速发展，空间开销过大
2. 用户历史数据过于稀疏，对于购买次数较少的用户来说，找到相似用户的准确度较低，不适于低频应用

2. **ItemCF**

1. 共现矩阵与UserCF相同，以用户为行向量，物品为列向量
2. 计算共现矩阵两两列向量之间的相似性，构建**物品相似度矩阵**
3. 获得用户历史行为数据中的正反馈物品列表
4. 计算物品相似度并排序

$$R_{u,p} = \sum_{h \in H}(w_{p,h} \cdot R_{u,h})$$

H是目标用户的正反馈物品集合，$w_{p,h}$是物品p和物品h的物品相似度，$R_{u,h}$是用户h对物品h的已有评分

3. **UserCF和ItemCF的应用场景**

UserCF社交性强，适用于新闻推荐场景（兴趣点分散，新闻具有及时性、热点性）

ItemCF稳定性强，适用于视频推荐、商品推荐场景（兴趣变化稳定）

4. **协同过滤的下一步发展**

    1. 泛化能力弱，不能将两个物品相似的信息推广到其他物品相似性上
    2. 头部效应明显，热门物品容易和大量物品产生相似性
    3. 解决方案：**矩阵分解算法**

矩阵分解

1. **原理**

    通过分解共现矩阵，为每一个用户和物品生成一个隐向量，将用户和物品定位到隐向量表达的空间上，距离相近的用户和物品最为符合

$$R_{m\times n}=U_{m\times k}V_{k\times n}$$

    k是隐向量的维度。k越小，隐向量表达信息越少，模型泛化程度越高。k越大，隐向量表达信息越多，模型泛化程度越低，求解复杂度越高

    已知用户矩阵U和物品矩阵V，用户u对物品i的预估评分为

$$\hat{r}_{u,i}=q_i^Tp_u$$

    p_u是用户u在用户矩阵U中对应的行向量，q_i是物品i在物品矩阵V中对应的列向量

2. **矩阵分解的求解过程**

    1. **特征值分解**

        仅适用于方阵，不适用于分解用户—物品矩阵

    2. **奇异值分解**

$$M_{m\times n}\approx U_{m\times k}\Sigma_{k\times k}{V_{k\times n}}^T$$

        缺点：共现矩阵要求稠密，但实际上非常稀疏；计算复杂度达到了$O(mn^2)$级别

    3. **梯度下降**

        矩阵分解的目标是使原始评分r_{ui}与用户向量和物品向量之积$q_i^Tp_u$的差尽可能小，即求

$$\min\sum_{(u,i)\in K}(r_{ui}-q_i^Tp_u)^2$$

        为了减少过拟合现象，加入正则化项后的目标函数为

$$f=\min\sum_{(u,i)\in K}(r_{ui}-q_i^Tp_u)^2+\lambda(\|q_i\|^2+\|p_u\|^2)$$

        其中λ是正则化系数

        对目标函数求偏导

$$\frac{\partial}{\partial q_i}f=-2(r_{ui}-q_i^Tp_u)p_u+2\lambda q_i$$

$$\frac{\partial}{\partial p_u}f=-2(r_{ui}-q_i^Tp_u)q_i+2\lambda p_u$$

        沿梯度反方向更新参数

$$q_i=q_i+\gamma((r_{ui}-q_i^Tp_u)p_u-\lambda q_i)$$

$$p_u=p_u+\gamma((r_{ui}-q_i^Tp_u)q_i-\lambda p_u)$$

        其中γ是学习率

        反复迭代直到达到迭代结束标准

3. **消除用户和物品打分标准的偏差**

在矩阵分解时加入用户和物品的偏差向量

$$r_{ui} = \mu + b_i + b_u + q_i^T p_u$$

μ是全局偏差常数，b_i是物品偏差系数，b_u是用户偏差系数

矩阵分解的目标函数更改为

$$\min \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2)$$

4. **矩阵分解的优势和局限**

优势：

1. **泛化能力强**，隐向量生成的过程是对共现矩阵进行全局拟合的过程，隐向量是利用全局信息生成的，有更强的泛化能力。
2. **空间复杂度低**，只需要存储用户和物品隐向量，空间复杂度由n^2级别降低到$(n+m)\times k$级别
3. **更好的扩展性和灵活性**，隐向量便于与其他特征进行组合和拼接，便于与深度学习网络结合

局限：

不方便加入用户、物品和上下文相关的特征，丧失了利用很多有效信息的机会（每一次加入新的信息都需要重新计算