基础知识

一次搜索行为的整体链路：

- 用户输入、引导词触发
- 前置操作: 搜索跳转，关键词预处理（大小写转换）
- query parse：纠错、意图、同义词、分词、实体识别
- 召回：
  - 文本召回(es 三级)、实体召回、向量召回(中间上过)
  - 广告召回、其它业务场景数据召回（源本、今日达、运营）
  - 召回同时会做业务过滤(限量和限售)
- 排序：ctr、相关性
- 重排： 业务规则排序调整
- 后置操作： 分页、获取商品详情、售罄沉底

基本指标

- GMV: 1.8亿
- pv：800万、uv：300万
- qps： 5000-8000 10台
- 响应时间：单纯ctr（10-12ms）或相关性进本30-35ms，都做了分批预测优化、相关性还做了缓存
- 接口耗时: 350ms
- pv点击率：78%、pv加购率：47%、pv转化率：35%
- 平均加购位置：5、平均点击位置：9、首条加购率18%
- 搜索词数量：14万、kwd_item数量：700万， 平均50个召回

召回

召回策略

- 兜底召回
  - 热门召回: 分运营区域实时统计，热销商品、热门分享商品、热门搜索商品
  - 年维度召回: 用户去年这个时间点前后N天购买商品召回今年的这些商品(二级类目粒度、实体粒度)
- 关联规则统计：
  - 固定搭配：根据用户点击召回搭配商品
  - 一个是核心需求，基于用户搜索行为链路，挖掘搜索后加购共现商品
  - 一个是扩招需求，基于用户加购行为链路，在时间窗口内共现商品
- 协同过滤：
  - itemcf、deepwalk、user cf
- 个性化召回
  - 复购召回：用户经常复购什么，召回什么
  - 历史搜索召回：用户历史上搜索词，召回关联的商品
  - 用户模型召回: 使用模型预测用户偏好二级类目，进行召回
- embedding召回：
  - 用户最近点击召回相似相关的商品
  - 历史30天top1商品相似度召回，query相似度，单塔、多塔
- 冷启动召回：挖掘点击曝光率少转化率高的商品，采用汤姆森采样排序挑选TOP商品，在首页瀑布流第4页第2个位置召回冷启动商品，进行流量扶持
- 社交推荐：基于社交网络中好友的购买行为，进行商品召回。例如，用户的朋友最近购买了某款电子产品，那么推荐该产品给用户。

dssm双塔有哪些改进：

- 缺点是由于，双塔模型结构比较简单，每个塔都是独立的一个dnn，底层输入隔离，只在上层的cosine做交互，但是这时参与交叉的user/item embedding，已经是高度压缩的了。一些细节信息已经损失，失去了与对侧信息交叉的机会，在训练和部署时采用“双塔分离”的结构，主要是为了保障线上快速响应的优点，也是他主要用在召回和粗排上的主要原因。
- 改进：
    - 模型
        - SENet的思路就是，在将信息喂入塔之前，插入SEBlock。SEBlock动态学习各特征的重要性，增强重要信息，**弱化甚至过滤掉原始特征中的噪声，从而减少信息在塔中传播过程中的污染与损耗**，能够让可能多的重要信息“撑”到final dot product那一刻。
        - resnet方式：将一些重要的、细粒度的信息也抄近路到中间以及最后一层。但是要慎重选择特征，不然那么final embedding就会膨胀好几倍
            - “极其个性化”（e.g., userId, itemId）的特征，和，对划分人群、物群有**显著区分性**的特征（e.g., 用户是新用户还是老用户？用户是否登陆？文章所使用的语言，等）
        - 多塔生成embedding：
            - 可以沿适合自己的塔向上流动浓缩，避免相互干扰。最后由每个小塔的embedding聚合成final embedding**，与对侧的final embedding做dot或cosine。
            - 并联双塔，通过并联多个双塔结构增加双塔模型的宽度，来缓解双塔内积的瓶颈从而提升效果，另外，涉及到并联双塔训练细节的是
                - 由于FM和DCN等结构，只能完成信息交叉，而无法信息压缩，所以只能喂入有限的重要特征，否则会引发维度膨胀。
                - **虽然不同结构可能共享特征，但是它们却不共享这些特征的底层embedding。**同一个特征，如果要同时喂入

MLP和DCN，就必须定义两套embedding，供MLP和DCN分别加以训练。根据我之前的经验，分离embedding空间的确能够换来性能上的提升，但是也带来模型膨胀，给线上serving带来压力。

- 在各自塔中增加自适应的user和item向量，《A Dual Augmented Two-tower Model for Online Large-scale Recommendation》：

- 负样本：Training Data Mining: 不能简单的使用曝光未点击样本做副样本，因为和线上数据不符
  - 全局随机负采样，对付样本采用word2vec汇总方式按出现频率进行采样进行负采样，可以保证冷门物料也能被抽样，热门物料被打压，α 一般设置0.75
  - 挖掘Hard Negative增强样本
    - Airbnb在《Real-time Personalization using Embeddings for Search Ranking at Airbnb》一文中的做法，就是根据业务逻辑来选取hard negative
      - 增加与正样本同城的房间作为负样本，增强了正负样本在地域上的相似性，加大了模型的学习难度
      - 增加“被房主拒绝”作为负样本，增强了正负样本在“匹配用户兴趣爱好”上的相似性，加大了模型的学习难度
    - 当业务逻辑没有那么明显的信号时，就只能依靠模型自己来挖掘。这也是本文与百度Mobius的作法，二者的作法极其相似，都是用上一版本的召回模型筛选出"没那么相似"的对(也就是召回中段的结果，一般easy：hard为100:1)，作为额外负样本，训练下一版本召回模型。
    - 不同难度的模型融合
      - 不同难度的模型独立打分，最终取Top K的分数依据是多模型打分的加权和（各模型的权重是超参，需要手工调整）
      - 串行融合：其实就是粗排，候选物料先经过easy model的初筛，再经过hard model的二次筛选，剩余结果再交给下游，更复杂粗排或是精排。根据文章中的

经验，使用“曝光未点击”作hard negative训练出来的hard model同样没有效果，反而是挖掘出来的hard negative训练出来的hard model做二次筛选更加有效。

- 内容：
  - 丰富每个塔的输入，用户兴趣，区域、年龄、用户行为序列
  - 物料：文本、tag、属性信息

推荐搭配购买，挖掘搭配商品

- **关联规则挖掘**：通过频繁项集挖掘来找到经常一起出现的商品对。
  - 核心需求：搜索行为链路中的二次搜索后加购
  - 拓展需求：比如时间窗口内的共现次数、上下文连续加购
  - 人工定义的一些关联规则：功能属性搭配
- **协同过滤**：
  - **基于用户的协同过滤**: 分析哪些用户的购买行为相似，如果某个用户购买了A商品，那么推荐与A商品在其他用户购买记录中经常一起出现的商品。对已购过滤，然后根据用户相似性，和用户对他的行为（打分、加购次数，是否购买）进行加权，倒排推荐
    - **权重并不是最大购买次数**: 这里的权重是用户相似度，而非购买次数。购买次数可能会被用作行为的一个强度指标（如评分），但权重本质上是相似度。
    - **处理冷启动问题**: 在推荐新商品或给新用户推荐时，相似度计算和行为数据可能不足。为此，可以结合基于内容的推荐或其他方法来弥补。
  - **基于物品的协同过滤**: 分析哪些商品经常被相同的用户购买，商品之间的相似度可以通过相似度度量（如余弦相似度）来计算。
- **图挖掘（Graph Mining）**
  - 将商品和用户看作图中的节点，用户的购买行为作为边。使用图的社团发现算法、随机游走、PageRank等方法来发现商品之间的潜在搭配关系。
- **上下文感知推荐**：

    - **用户意图理解**: 在搜索场景中，根据用户输入的关键词，结合上下文理解用户当前的需求。例如，用户搜索“运动鞋”时，可以理解为用户可能需要与运动相关的产品，推荐相关的运动服装、袜子等。
    - **动态组合推荐**: 通过使用深度学习模型（如DIN或Wide & Deep）来捕捉用户的短期和长期兴趣，并结合当前搜索意图，推荐相关的搭配商品。
- **实时搭配推荐模型**
    - **基于深度学习的搭配推荐**: 构建一个端到端的推荐模型，例如使用双塔模型（Dual Tower Model），一边输入用户的搜索意图，另一边输入商品信息，模型通过训练能够学习到哪些商品在一起更容易被用户接受。
    - **个性化推荐**: 考虑用户的历史行为和个人偏好，在推荐搭配商品时，根据用户画像进行个性化处理。
- 排序侧的措施：
    - 特征：
        - 基于商品特征，
            - **商品相似度**: 根据商品的属性（如品牌、类别、价格等）计算相似度，并将相似度作为特征，融入排序模型中。
            - **搭配频次**: 统计商品在历史订单中作为搭配的频次，作为排序特征之一。
        - 用户行为特征
            - **用户购买历史**: 根据用户历史购买记录生成特征，识别用户偏好的搭配商品。例如，用户曾购买的商品可能会影响推荐的搭配商品。
            - **点击行为**: 用户点击某个商品时，可以考虑推荐与该商品常被一起购买的商品。
        - **上下文特征**
            - **搜索意图**: 用户的搜索关键词可以帮助理解其当前需求，从而在排序时更优先考虑符合搜索意图的搭配商品。
            - **时间和地点**: 结合用户的时间和地点数据，推荐适合当前场景的搭配商品。例如，季节性商品或地方特色商品。

- 模型
  - 综合排序模型
    - **传统排序模型**: 如Logistic Regression、Gradient Boosting Trees（如LightGBM、XGBoost），通过特征工程将搭配商品的相关特征引入模型中进行排序。
    - **深度学习排序模型**: 使用深度学习模型（如Wide & Deep、DeepFM）捕捉复杂的特征交互，提升搭配商品的排序效果。
  - 搭配推荐特定模型
    - **双塔模型（Dual Tower Model）**：一边处理用户特征，一边处理商品特征，模型通过学习用户和商品之间的匹配度来进行排序。在这种模型中，可以将搭配商品作为额外的输入信息，提升匹配效果。
    - **多任务学习（Multi-task Learning）**：同时优化多个任务，如主商品推荐和搭配商品推荐，通过共享模型参数来提升整体排序效果。
    - **强化学习（Reinforcement Learning）**：使用强化学习算法（如Deep Q-Networks）来优化排序策略，根据用户反馈（点击、购买等）来调整排序策略，提升推荐的准确性。
- 排序优化
  - 个性化排序
    - **用户画像**: 利用用户画像信息进行个性化排序，使推荐的搭配商品更符合用户的个人偏好。
    - **历史行为加权**: 根据用户的历史行为对搭配商品进行加权，提高推荐的相关性和个性化程度。
    - 价格偏好
  - 实时动态调整
    - **实时数据**: 根据实时的用户行为数据动态调整排序，例如，当用户频繁浏览某种商品时，相应的搭配商品可以优先展示。

        - **A/B 测试**: 实时监控不同排序策略的效果，通过A/B测试不断优化排序模型
    - 上下文相关排序
        - **上下文适配**: 在搜索结果中，根据当前上下文信息（如搜索词、用户意图等）对搭配商品进行排序，以提供更贴合用户需求的推荐。
- 案例:假设用户搜索“夏季连衣裙”，排序模型可能会：
    1. **分析搜索意图**: 识别用户可能需要夏季相关的搭配商品。
    2. **计算搭配商品的相似度**: 推荐与连衣裙搭配度高的商品，如夏季凉鞋、配饰等。
    3. **个性化排序**: 根据用户的历史购买记录和偏好对搭配商品进行排序。
    4. **实时调整**: 根据用户在页面上的行为（如点击、浏览时间）动态调整推荐结果。

基于策略的排序或者召回

- 召回：
    - 关联规则：固定搭配、
    - 统计：热销、复购

冷启动

基于内容，特征共享

- 通过更多的属性信息，共享特征
- 用户冷启动：
    - 基本属性: 如年龄、性别、所在地、工作行业
    - 行为引导、外部数据兴趣标签：通过问卷调查、初始选择或从社交媒体账户导入兴趣标签。根据这些标签推荐相应的商品类别或特定商品。
- 商品冷启动：

- **商品属性匹配**: 使用商品的属性（如类别、品牌、价格、颜色、功能、实体等）与用户的偏好或历史行为进行匹配。例如，一个新上架的电子产品可以推荐给那些经常购买或浏览类似电子产品的用户。
- **文本分析**: 对商品的描述、标题或用户评论进行自然语言处理（NLP），提取关键特征或主题标签，与用户的兴趣标签进行匹配。
- 多模态，文本图像

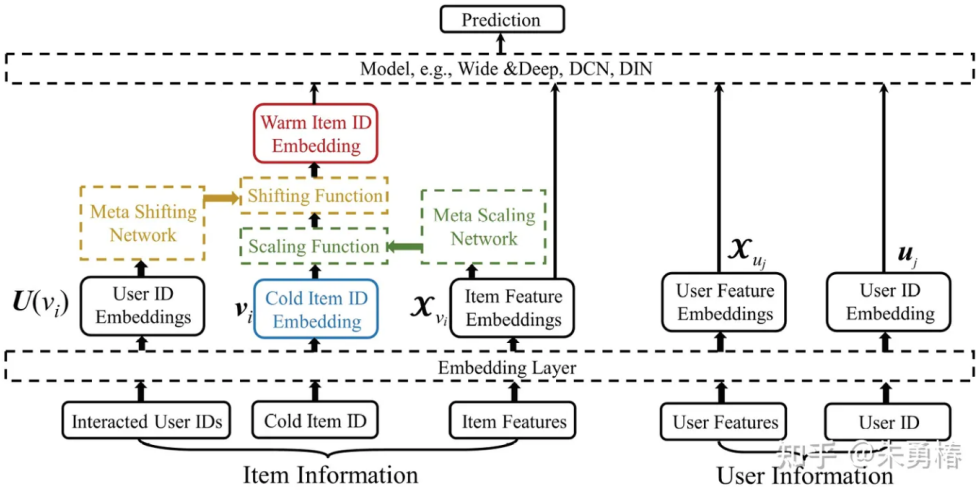
user embedding生成方式：

- 始化的基底分解，新用户或商品的embedding表示，通过集中属性信息的embedding，进行压缩，这样的embedding经过更少的行为数据就能收敛到一个比较好的位置。
  - MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation，本质上还是从user profile间接得到用户表示
- 直接生成新的embedding，直接建立网络从user profile生成User ID embedding
  - melu：
    - （1）根据support set上的损失进行局部更新时，不更新embedding的参数，目的是为了确保学习过程的稳定性。并且，这里是想让模型做一个假设，假设users和items不变，改变的是users的想法。（这里很有意思，support set只更新NN的权重，这样在query set上，代表用户的user embedding和代表item的item embedding与support set相同，而值更新nn的参数，计算user item相关度的模型参数变了，根据support set计算的用户的偏好被NN部分保留了下来。
    - （2）同样，与MAML不同的是不限制support set的规模。
    - （3）根据query set上的损失，对所有参数进行更新。这里定义support set上的损失为用户个性化的参数更新，根据

query set上的全局更新来根据小部分数据为模型选择更好的参数，

- MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation[1], 本质上还是从user profile间接得到用户表示
- SIGIR2019的Warm Up Cold-start Advertisements: Improving CTR Predictions via Learning to Learn ID Embeddings, 核心是训练一个生成器，输入profile特征，输出ID的embedding。提出了基于meta learning的有效的方法，该方法使用第二步梯度更新模型参数。为新的广告学习一个更好的初始embeddind，是第一个同时优化冷启动和warm-up两个任务的有监督框架。
  - Better at cold-start：对于冷启动的广告，可以获得一个可接受的预估结果，即模型对于没有出现在训练集中的广告id，有一个较小的损失。
  - Faster at warming-up：对于长尾的广告（有小部分样本），模型可以快速收敛适应这样的id。
- WUMF：Learning to Warm Up Cold Item Embeddings for Cold-start Recommendation with Meta Scaling and Shifting Networks， 2021年
  - 冷启动的两个问题
    - **冷启动ID embedding和深度模型之间存在一个gap**。在推荐系统中，存在一个众所周知的现象，少量热门物品占据了大量交互样本，而大量冷启动物品只有少量交互数据。深度模型是在所有数据上进行训练，因此这个模型会学到大量来自热门物品的知识。而冷启动ID embedding仅仅在特定物品的交互数据上训练，因此冷启动ID embedding很难拟合这个深度模型。所以针对冷启动物品如何加速模型的拟合是至关重要的（**fast adaptation**）。
    - **冷启动物品ID embedding会受到噪声严重的影响**。在推荐系统中，随时都在出现错误的交互，比如错误点击。而物品ID embedding完全取决于该物品的交互数据。而冷

启动物品通常只有很少的交互数据，因此很小的噪声也会对冷启动物品ID embedding造成严重的影响。

- 更好的初始化商品的embedding，
  - **Common Initial Embeddings**: 我们提出了一种公共初始化embedding，使用现有的所有物品的ID embedding均值作为新加入的物品的初始化embedding。
  - **Meta Scaling Network**：我们希望将冷启动物品ID embedding转换到一个更好的特征空间，能更好地拟合深度模型。对于每一个物品，冷启动ID embedding和warm ID embedding之间存在一定的联系，我们认为相似的物品，这个联系也应该相似，因此Meta Scaling Network以物品的其他特征做为输入，输出一个拉伸函数
  - **Meta Shifting Network**: 所有交互过的用户均值可以有效的减轻异常用户的影响，因此Meta Shifting Network以交互过的用户的均值表示作为输入，输出一个偏移函数：
  - 如果把一个item embedding表示为与它交互过的user embedding的平均，可以达到滤除离群点的效果。根据这一点可以用交互过的user embedding输出一个偏移量，使item embedding更稳定一些。这里我可以理解为每一类用户在空间中是分开的，因此对准到某一类用户身上就是加一个shift的操作。
  - 

  -

整体框架如上所示，深度模型的输入包含4个部分，物品ID embedding，物品其他特征embedding，用户ID embedding，用户其他特征embedding：

$$\hat{y} = f(\boldsymbol{v}_i, \boldsymbol{\mathcal{X}}_{v_i}, \boldsymbol{u}_j, \boldsymbol{\mathcal{X}}_{u_j}; \theta),$$

Common Initial Embeddings: 我们提出了一种公共初始化embedding，使用现有的所有物品的ID embedding均值作为新加入的物品的初始化embedding。

Meta Scaling Network：我们希望将冷启动物品ID embedding转换到一个更好的特征空间，能更好地拟合深度模型。对于每一个物品，冷启动ID embedding和warm ID embedding之间存在一定的联系，我们认为相似的物品，这个联系也应该相似，因此Meta Scaling Network以物品的其他特征做为输入，输出一个拉伸函数：

$$\boldsymbol{\tau}_{v_i}^{scale} = h(\boldsymbol{\mathcal{X}}_{v_i}; w_{scale}),\ \boldsymbol{\tau}^{scale} \in \mathbb{R}^k$$

Meta Shifting Network: 所有交互过的用户均值可以有效的减轻异常用户的影响，因此Meta Shifting Network以交互过的用户的均值表示作为输入，输出一个偏移函数：

$$\boldsymbol{\tau}_{v_i}^{shift} = g(\mathcal{G}(\boldsymbol{U}(v_i)); w_{shift}),\ \boldsymbol{\tau}^{shift} \in \mathbb{R}^k$$

最后转换后的物品ID embedding表示为

$$\boldsymbol{v}_i^{warm} = \boldsymbol{v}_i \odot \boldsymbol{\tau}_{v_i}^{scale} + \boldsymbol{\tau}_{v_i}^{shift}$$

整个训练流程分为两个阶段，第一个阶段训练推荐模型，第二个阶段训练两个meta network，流程如下所示：

- 用户分层，使用预训练的group embedding，在正式训练时进行冻结，再加上正常的 user personal resdual embedding，最为最终的用户embedding表示
  - group embedding预训练，在我的实践中，进行以上预训练：

    - 分群是“初始用户群”，划分时，只能依靠那些对新用户友好的特征，比如：年龄、性别、安装那些（种类）的app，是被哪个（类）广告拉新进来的、....
    - **只采用新用户的数据**，不用老用户的数据这么做是因为，老用户交互的item，主要是由推荐系统根据用户历史推荐出去的，和用户的初始元信息（比如，年龄、性别、App等）关系已经不大了。引入老用户数据，反而引入了噪声
  - 正式训练：

- $E_{InitGroup}$**对于老用户要stop gradient**，毕竟年代久远，init user group embedding不应该再为老用户推荐结果上的loss负责。也防止规模巨大的老用户数据，将已经训练好的$E_{InitGroup}$带偏
- **至于新用户，最好对**$E_{InitGroup}$**也stop gradient**，一来新老用户一致，方便实现；二来，我们希望信息都积累在$E_{PersonalResidual}$上，而不是在group embedding上。

模型角度：

- 直接两个不同的模型去训练，参数量多，另一个是训练样本少，比较难训练出来一个好的效果
- POSO：Personalized Cold Start Modules for Large-scale Recommender Systems，共享参数，同一个模型，但是对特征进行mask
  - **选择指向性强的特征，能够明确的区分新老用户。** uid 不一定能cover住，首先它不是明确的区分新老用户的关键特征，也就是说指向性不强，你也不太清楚它会学成啥样。
    - 一般深度模型都是id类特征是一等公民，所以常见的类似vv特征都会做分桶处理，然后查embedding做不断的调整。但是这次真的不一样，你的输入就是一个数值特征不需要做embedding变换。这里所说的不要做变换是指不需要进行embedding处理，但是vv本身需要一些简单的处理，作者在他的知乎文章里补充了高斯有界化处理，那接么下来我们就说下这个高斯有界化处理注意事项。
      - 高斯函数是一个 0 均值，至于为什么是 0 均值，可以这么理解。我们的vv数都是大于等于0的，我们只要关注高斯右半边即可。这样处理之后数值会相对平滑一点，相比于直接拍下脑袋说大于150vv就是老用户而言显得更好一点。
      - 方差如何选择？这个是一个超参，根据你自己的业务实际的情况调整。这里需要说明的是，方差越大倾向于老用户，方差越小倾向于新用户，所以最佳数值交给AB测试吧！

排序

观测指标

- gmv、qty、pv点击率、pv加购率、点击率、加购率、topk点击率、加购率、日活200w、平均召回50
- 纬度：省区、城市、keyword分层下的指标、用户分层
    - 不同区域商品的搭配购买习惯是不通的
    - 高中低频的搜索词下，用户的表现差别是比较严重的。

如何确定迭代方向

- 竞品调研，看看业界排序是如何迭代的
- 运营反馈
- 阶段性业务需求：比如好评优先，新品提权、大份装的提权与降权、毛利率
- 自查：通过分析用户行为数据进行badcase 挖掘以及问题归因，比如
    - keyword整体加购率、top位置加购率的对比
    - 商品加购率很高，但是平均加购位置很靠后
    - 搜索后并未在搜索场景加购、但在平台加购
    - level很低加购率高、level高加购率低等、或者是es分数
    - 点击率大幅上升，但转化率不升反降

难点：搜索排序个性化和相关性是如何融合的

- 搜索排序中最大的难点就在于怎样融合相关性与ctr的关系，在这个方面也做了很多尝试
    - 首先是增加相关性特征，比如es召回的粗排分数，向量相似度，以及文本匹配特征
    - 然后是根据一些特征对样本进行调权，比如相似度较高且公共子串长度较长的加权
    - 或者是模型侧增加相关性约束loss，增加正样本的相关性loss

- 但是单一模型这些操作，只能进一步的改善相关性问题，并不能解决，有两方面原因吧
    - 上述方法本质上都是加强相关性的一些操作，但是ctr的目标还是点击率预估，一旦用户行为超过某个界限，就会导致相关性出现问题，而且线上服务一旦出现badcase，考模型更新解决周期较长
    - 由于业务场景本身的特点，采用保召回的策略，叠加平台用户的复购需求较为强烈的时候，很难避免相关性问题，所以需要对相关性有一个明显的分界

有没有模型结构的优化，是采用业界的标准解法吗

- 模型结构的优化不多，只有两种吧
    - 一种是添加约束，也就是辅助损失，比如相关性分层训练过程中，会出现模型预测输出摇摆的问题，类别最大是3，然后就是0，跨度太大，也表明模型对学到的信息不置信，所以添加了一个约束，就是标签距离近的概率要大于标签远的概率，而且直接使用logits的输出去优化，更容易收敛一些。
    - 另一个就是对于商品的冷启动，增加一个mask模块，选取能够区分冷门商品特征，通过一个attnetion模块对（sigmoid + fc的模式），对输入做mask，训练一个相当于门控的单元，实现特征提取的分化，冷门商品应该关注哪些特征，热门商品应该关注哪些商品。

冷门样本稀疏怎样处理

- 数据角度
    - 数据增强：通过引入外部数据来丰富冷门样本的信息。例如，从外部获取商品的描述、用户评价、品牌影响力等信息来补充冷门商品的特征。
    - 关联泛化特征：对特征粒度进行聚合，类目、实体、品牌，丰富冷门商品的用户交互行为

    - 基于相似行的扩展：对相似的商品的行为和数据进行扩展
    - 协同过滤：通过协同过滤算法（如基于邻域的方法或矩阵分解）来填补冷门样本的行为数据缺失。这些算法可以根据商品或用户的相似性推测出冷门样本的潜在行为。
    - **图神经网络**：利用图神经网络（Graph Neural Network, GNN）建模商品之间的关系和用户行为。通过GNN，可以将冷门样本与热门样本连接起来，进行信息传播，从而丰富冷门样本的特征表示。

      知识图谱：构建商品和用户的知识图谱，将冷门样本纳入到图结构中，利用图中的连接关系和属性信息来补充冷门样本的数据。
- 样本
    - **过采样冷门样本**：通过数据增强技术（如SMOTE）来增加冷门样本的数量，使其在训练集中占有更大的比例。
    - **欠采样热门样本**：对热门样本进行欠采样，减少其在训练集中的数量，平衡热门与冷门样本的比例。
    - **数据重加权**：根据样本的流行度分配不同的权重，在训练过程中给予冷门样本更高的权重，确保模型在训练时对冷门样本的关注度。
    - **启发式**的方式来扩充正样本。通常是基于经验或观察而设计的策略，用于解决某些问题。为了缓解正样本稀疏的问题，启发式方法可以用来扩充正样本，通过各种策略生成或挖掘更多相关样本。
        - **基于行为的相似性**
            - **相似用户行为**：对于一个用户的正样本，找到行为相似的其他用户的正样本。例如，如果用户 A 和用户 B 对某些商品有相似的行为模式，可以将用户 B 对某商品的点击视为用户 A 的正样本。
            - **相似商品推荐**：根据商品的特征（如类别、品牌、价格）找出相似商品。如果一个用户对某商品表现出兴趣，则可以将相似商品作为正样本进行推荐。
        - **基于内容的匹配**

        - **内容特征**：根据商品的内容特征（如描述、标签、关键词）生成正样本。例如，如果用户对某个关键词的商品表现出兴趣，可以扩展到与该关键词相关的其他商品。
        - **相似内容**：通过内容匹配算法（如余弦相似度、Jaccard相似度）找到内容相似的商品，并将这些商品作为正样本。
    - **利用现有样本的相似性**
        - **样本扩展**：利用已有正样本的特征生成新的正样本。例如，通过对已有正样本进行特征扰动或数据增强，生成类似的正样本。
        - **数据增强**：通过数据增强技术（如噪声注入、特征变换）生成新的正样本，以扩展正样本的数据集。
- 模型
    - 冷启动模型：设计专门的冷启动模型，用于处理冷门商品或新商品。在冷启动阶段，可以使用基于内容的推荐或基于规则的推荐，以弥补行为数据的不足。
    - **元学习**：通过元学习（Meta-Learning）技术，使模型能够快速适应冷门样本。元学习可以让模型通过少量训练样本快速学习到冷门样本的特征，提升其在冷门样本上的表现。
- 策略
    - 集成学习：利用多种模型的集成方法，如基于规则的推荐、基于协同过滤的推荐、基于内容的推荐等，混合不同模型的结果，以提高冷门样本的召回率。
    - **混合推荐**：根据用户的偏好和商品的冷门程度动态调整推荐策略，例如对新用户或冷门商品更多依赖基于内容的推荐，而对有丰富行为数据的用户更多依赖基于协同过滤的推荐。

热门商品的长尾效应怎样避免

- 问题：电商平台通常拥有大量的商品，其中有一部分是长尾商品，即销量较低但种类繁多的商品。如何平衡长尾商品和热门商品的排序，让用户既能发现新颖的长尾商品，又能看到热门的高销量商品，是一个挑战。

- 引入探索机制，适当增加对长尾商品的展示机会。
  - ε-贪婪策略：在排序结果中，按一定概率（ε）随机插入一些长尾商品，以增加它们的曝光机会。
  - 上下文多臂老虎机（Contextual Multi-Armed Bandit）：结合用户的实时行为和历史数据，动态调整推荐策略，既考虑热门商品的点击率，也增加长尾商品的展示机会。
  - 动态调控曝光机制
    - 曝光限制：对热门商品设置曝光限制，控制它们在某个时间段内的最大展示次数。
    - 曝光激励：对长尾商品设置曝光激励，增加它们在某个时间段内的展示次数。
- 重排多样性控制

怎么评价排序效果，验证有效性

- 首先就是明确优化的目的是什么，比如长尾词下的排序效果，首先是整体的大盘指标必须是正向gmv、ctr等。但是像这种目标优化，罗岛整体上的收益肯定不明显，因为整体上都是28分布，就需要去对query按pv分层的去看具体效果。来验证有效性。看是否起到相应效果，有没有降低正常词的一个效果
- 上线前也能够通过auc、gauc以及对历史数据的回溯，一定程度上感知模型的有效性

在没有abtest时候怎么显示你的一个收益

- 离线评估预期、上线后主要观测指标的提升和预期是否相近

相关性badcase挖掘

- ctr：位置和点击率拟合曲线，偏离严重的点
- **用户反馈**：建立一个机制，让用户对搜索结果进行评价。如果用户认为某个结果不相关，可以将其标记为badcase。
- **行为日志挖掘分析**，ctr很高但是level很低， level很高，但是明显低于其他level的ctr，

- **es召回对比**，es召回分数很高，但是level很低
- **不同版本对比**：新版本在某些query上明显低于老版本
- **长期监控**：建立一个监控系统，定期检查搜索结果的质量，自动挖掘可能的badcase。

字节的相关性场景，样本太多了，怎么去自动的挖掘相关性样本

- **规则引擎**：构建规则引擎，根据设定的内容匹配规则和后验数据，自动筛选相关性样本。这些规则可可以是基于业务需求定制的规则。
    - 规则：根据定义的相关性层级，以及标注体系，去定义一些显示规则，比如在我的场景，根据对搜索词进行解构，能够定位到相对的层级，搜苹果，最高等级就是实体命中为2，龙牌酱油那最高就是3，那么在结合一些后验数据和文本的匹配规则，去辅助打标
- **半监督学习**：结合有标注的数据和大量未标注的数据，使用半监督学习方法来提升样本标注的效率和准确性。
- 结合规则引擎和半监督学习，挑选出置信度较高的样本加入到训练集
- 数据增强：通过数据增强技术生成更多的训练样本。
    - 人工构造样本，根据item内容结构，进行样本构造
    - 例如，通过对现有样本进行扰动或变换来生成新的相关性样本。
- 对比学习
    - 对比学习通过生成和优化样本对来学习数据的相似性和差异性，在自动化收集相关性训练样本方面具有重要应用价值。它可以提高样本质量、优化数据标签、增强推荐系统和处理冷启动问题。然而，应用对比学习时需要解决样本选择、计算成本和模型设计等挑战。通过合理设计和优化，可以有效利用对比学习技术提升相关性样本的自动化收集效果。

线上效果不及预期怎么解决

- 数据验证：确保特征数据准确，无异常值、数据处理流程是否一致
- 确定模型是否正确：少量样本、较大的学习率，是否能过拟合

- 交叉验证：使用K折交叉验证方法，评估模型的泛化能力，是否过拟合。
- 查看模型的各部分输出，看是否异常（比如样本不均衡，最终的logitis输出明显两极化）
- 分析特征重要性，通过case对比，分析新增特征的影响，是否需要进一步加工（归一化、作差）
    - 比如好评商品优先，好评率的分布集中在80%-95%，没有一个明显的差距
    - 但是如果转换一下，1-好评率，会使差距变得明显

如何保证服务的稳定

- 监控
    - 指标：波动范围、突变（平均加购位置、首条加购率、加购率等）
    - 特征：异常值（空值）、数量在正常波动范围内
    - 服务：服务异常告警、次数、时间分布
- debug链路
    - 能够迅速查询整个ctr链路的全部输出、分析定位问题
- 人工干预链路：通过人共运营干预排序，迅速响应
- 兜底策略：相关性粗排

模型训练时是否是分布式

- 为什么要用分布式，结合业务需求
- 日更模型
- 数据量
- 数据并行、模型并行、参数更新

特征

排序特征是如何做的

- 首先是要建立一个比较清晰的特征框架体系：user、query、item以及两两之间的交互，然后就是根据业务的迭代，逐渐的去完善特征
- 至于每次迭代特征是如何做的，主要包括以下几个步骤，
    - **首先就是理解业务需求、识别相关特征**：也就是要考虑问题是什么、能抽象出什么共性、有哪些特征能够有助于完成本次迭代的目标，举几个例子
        - 比如说好评商品优先：比如商品、品类以及一些交互特征（是否好评、差评，评价数量）
        - 如何提高长尾词下的排序效果，那么首先要分析长尾词有什么特点：
            - 因为长尾词pv小，也就导致有效用户行为偏少且随机性比较大，只有个别商品有用户行为：
                - 如何建立商品之间的联系，就比如可以对搜索词下用户对tag、品牌、实体、类目的用行为进行关联汇总
            - 用户行为少就会使意图不准、就会导致较多的误召回，也就会出现相关性存在不准确的问题
                - 加购最多的tag、实体相关性要明显较强
                - 相关性分数、文本匹配、query长度等信息
    - **数据预处理**：对数据进行收集探索，去除异常、重复值和填补一些缺失值，对数据进行清洗，保证数据的质量，并使用可视化和统计分析（均值、方差、标准差，偏度、峰度）等方式查看数据的分布以及特征的性质（稀疏、稠密）。
    - **特征生成**：提取基本特征，并构造一些潜在组合特征、聚合特征、交互特征等
    - **特征转换**：编码、归一化、标准化以及特征交叉
    - **特征选择**：使用特征重要性或者与label相关性过滤方式进行特征的粗筛。
    - **特征评估与优化**：根据特征重要性以模型的性能对特征进行评估与优化

怎么去挖掘商品侧特征

- 商品内容：文本、图片
  - nlp（fasttext、w2c、bert、主题提取、关键词）、cnn、ResNet、多模态clip
- 属性：类目、品牌、实体、tag、价格
  - 实体识别、类目体系
- 质量：评论、收藏、复购率
- 交互特征：用户与商品的交互数据（点击、收藏、加入购物车、购买等），构建商品的交互特征，如点击率（CTR）、转化率（CVR），商品与搜索词之间的交互

做了哪些特征交叉、怎么做

- 特征交叉的本质就是去捕捉特征之间的复杂线性关系
  - 特征的共现：即特征值之间的联合出现情况。这种方式通常用于捕捉特征之间的组合效应，特别是在处理稀疏特征和高维特征时。特征的共现可以通过以下方式来理解
  - 向量相似度：在某些特定的情况下，特征交叉可以看作是通过计算向量相似度来捕捉特征之间的关系。例如，在深度学习模型中，特征交叉可能通过将特征嵌入到高维向量空间中，并计算向量的内积或其他相似度度量来实现。
- 交叉特征
  - 性别和类目、是否好评和是否购买，pv和ctr，加购来源份额占比和ctr
- 交叉的方式
  - 数值特征：简单+-*/、离散化转embeding做拼接或者内积，外积，FM
  - 类别特征：
    - 特征组合编码：是否好评和是否购买，00， 01， 10, 11
    - 特征编码直接拼接
    - hash分桶降维

        - 嵌入层（Embedding Layers），拼接、内积，外积

特征交叉的本质其实就是共现，个人经验：

- 特征分组的小技巧：交叉前要同质化，交叉时要差异化
- 类型相似的特征，可以放在同一个field里面，比如user的城市，年龄，性别，这些都是静态的，短时间内不会变化，就可以放在一起
- field之间尽量有差别，比如第一个以用户静态特征为主题，第二个field可以以用户兴趣标签之类的为主题，第三个可以选到作者侧上
- 交叉的时候倾向于user交叉item，在个人的经历中，item或者user自己内部的交叉收益不太大，而user和item的共现（co-occurrence）更加重要

如何保证数据的质量

- 排序序算法的性能和效果很大程度上依赖于输入数据的质量和标注准确性
    - 准确性
        - 数据源稳定可靠、定期验证
        - 数据清洗：去除或修正脏数据、重复数据、缺失值和异常值
        - 业务逻辑校验：比率>1，1d_cnt > 7d_cnt, 必须有曝光才能有加购
        - 标注数据的多轮标注、投票与审核
        - 监控
            - 数据条数、异常值
            - 特征的监控：**数据分布是否发生明显漂移**
    - 质量
        - 数据转换：标准化和归一化、分箱、log、标准化和归一化
        - 相关性分析、特征选择、特征重要性评估

偏态分布与正态分布的特征应该如何进行处理?

- 正态分布的特征，主要通过标准化和归一化处理；
- 对于偏态分布的特征，则通过对数变换、平方根变换、反向变换等方法，使其分布更接近正态分布，从而提升模型效果。根据具体业务场景和数据特征选择合适的方法，能够显著提高模型的性能和稳定性。

多样性

如和优化搜索结果的多样性

- **多样性的分类**：
  - 类目多样性
    - 类别平衡：根据商品的类别分配展示比例，避免某一类别的商品过于集中
    - 子类别多样性：在每个大类别下，进一步确保子类别的多样性展示
  - 基于属性的多样性
    - 品牌多样性：确保搜索结果中展示多个品牌的商品，避免单一品牌的商品占据多数。
    - 价格区间多样性：在搜索结果中展示不同价格区间的商品，满足不同消费层次用户的需求。
    - 颜色和款式多样性：根据用户的偏好，展示不同颜色和款式的商品。
  - 基于规则和模型的多样性
    - **规则引擎**：使用规则引擎定义多样性策略，例如：每页展示的前N个商品中必须包含至少M个不同类别的商品。
  - 用户行为：
    - **个性化推荐**：根据用户的历史行为和偏好，推荐不同类型的商品，增强搜索结果的多样性。
    - **相似商品推荐**：在搜索结果中插入一些相似但不同的商品，增加用户的选择范围。
- **多样性的评估**：

    - **多样性指标**： 类别覆盖率、品牌覆盖率
    - **用户反馈**：通过用户反馈了解多样性展示的效果，及时调整策略。
- **实现方法**：
    - 重排：在传统的相关性排序基础上，加入多样性因素，通过优化排序算法实现多样性。例如，可以使用贪心算法在保证相关性的同时，插入不同类别的商品。
    - 平滑技术：通过平滑技术，确保在一个固定窗口内，不同类型的商品均衡展示。例如，使用平滑因子对不同类别的商品进行重新排序，使得每个类别的商品均匀分布在搜索结果中。
    - 随机插入：在高相关性的商品之间，随机插入一些不同类别或属性的商品，增加搜索结果的多样性。这种方法可以避免单一类型商品的集中展示。
- **实际案例**：假设用户在搜索“手机”，为了实现搜索结果的多样性，可以进行如下处理
    - 类别多样性：展示不同品牌的手机，如苹果、三星、小米等。
    - 属性多样性：在展示不同品牌手机的同时，确保价格区间的多样性，如低价、中价和高价手机都有展示
    - 个性化推荐：根据用户的历史搜索和购买行为，推荐用户可能感兴趣但不同品牌和价格的手机。
    - 规则引擎：定义规则，确保每页展示的手机品牌不少于5个，价格区间覆盖至少3个档次。

相关性模型

- ernie：框架为paddle的ernie，
    - 蒸馏：
        - xbase：20-layer, 1024-hidden, 16-heads, 296M parameters. 30-45ms.
        - ernie-3.0-nano-zh：4-layer, 312-hidden, 12-heads, 18M parameters. 响应时间超100ms.
        - tempurature：hard lable， 相当于温度为1
        - 模型只有xbase的1/6

- huggingface：
  - bert预训练

理解

召回和排序区别

- 功能目标不同：
  - 召回：从一个大型的候选集合中快速筛选出一组与查询或用户兴趣相关的候选项。特点如下
    - **覆盖率**：广泛筛选，使用较为宽松的条件，快速地从大量的候选项中筛选出一部分相关的候选项，通常是数百到数千个
    - **高效性**：召回阶段强调速度，使用的模型和算法通常较为简单，计算量小，以便快速处理大规模数据。
  - 排序：对召回阶段筛选出的候选项进行精确排序，确保最相关或最优的项排在前面。
    - **精排**：使用更为复杂和精确的模型，对召回的候选项进行细致的打分和排序。
    - **高精度、高准确**：排序阶段强调准确性和相关性，通常使用更复杂的模型和特征来进行预测和排序。
- 也可以这样理解：召回也是一种排序，只不过这种排序我们使用的是多路简单的特征进行相似度的初筛，理由也很简单，我们在召回要尽可能的块；同时这样也提供了非常强的解释性。而粗排精排则为了更精确的给出排序结果，所以我们需要进行进一步的丰富特征或者使用不同的模型去更好的提取特征。

搜索与推荐的区别与共同点

- 区别：
  - 主动权在谁手里：推荐是用户被动接受、而搜索是用户主动产生的结果
  - 容忍度：推荐的容忍度更高，最多推荐的商品用户不感兴趣，而搜索有明确的相关性要求，很容易被看出来相关性的问题

    - 算法策略：推荐侧重于对用户行为的预测，强调的是个性化和探索性，搜索首要任务是找到用户查询最相关的东西，强调查询和结果之间的相关性，然后才可以考虑个性化和探索
- 共同点：
    - 都是通过用户对展示结果的点击、加购、购买行为反馈进行策略的优化

社区团购和传统电商存在区别

- 商品在售数量远远小于传统电商，由于保召回的策略，搜索结果存在较多的误召回
- 当天商品库变动很小、当天新增商品很少
- 客单价低，单次购买种类多，导致在一次搜索行为内，会加购多个商品
- 用户的加购意愿比较强烈，高于点击率10个百分点，且购买频次较高、复购行为明显
- 转化率高、延迟转化比例很低

收获

你觉得自己最有成就感的一个项目是什么，现在回过头去，你觉得还能怎么做的更好

- 对于我来说，其实我觉的我最满意的不是某一个项目，因为他是每次迭代必须要去解决的。而是总结了自己一套方法论，对于迭代需求应该怎么去抽象问题进行拆解，可以通过哪些方式解决问题、数据、样本还是模型（长尾词的效果），偶尔出现的优化结果不达预期（好评优先），线上线下不一致，怎么去分析定位哪里出了问题，通过什么方式解决。方案的可扩展性、怎么去评估线上大概率能生效。

开放问题

特征淹没

指的是在推荐系统中，当新用户或新物品加入时，系统试图利用大量的特征来描述这些新实体，但由于缺乏有效的互动数据，这些特征要么不具代表性，要么相互之间的重要性难以区分，导致模型难以从中提取有价值的信息。

MLP中激活函数为什么不用sigmoid,有什么问题么

- 梯度消失问题：在 Sigmoid 函数的输出接近 0 或 1 时，导数趋向于 0。这意味着在反向传播过程中，梯度可能会迅速衰减，从而导致前面几层的参数更新非常缓慢。这种现象被称为**梯度消失**，它会使得模型在训练过程中难以有效地更新参数，特别是在深层网络中，这个问题尤为严重。
- 计算复杂度: 相较于 ReLU 等简单函数，Sigmoid 的计算相对复杂。在计算 Sigmoid 函数及其导数时，由于涉及到指数运算（e^x），计算开销相对较大。在深度学习中，尤其是处理大规模数据时，这种计算成本会显著影响效率。
- 输出非零中心（Non-zero-centered Output）: Sigmoid 函数的输出范围是 (0, 1)，这意味着激活值始终为正数。这会导致在反向传播时，梯度的符号也总是同一方向。这种情况会导致梯度更新时可能出现某些特定的方向性偏差，影响参数更新的效率，减缓收敛速度

分类任务为什么不用MSE,从收敛的角度讲一下

- 梯度不适宜: 在分类任务中，输出通常为概率分布（如 softmax 输出），而目标值为0或1的独热编码（one-hot encoding）。使用 MSE 会计算预测概率与目标值之间的平方误差，
    - 由于 MSE 对于接近目标的预测输出产生的梯度较小，模型在学习过程中更新缓慢。尤其是当网络预测接近正确类别时，损失下降较慢，影响模型的收敛效率。
    - 而交叉熵损失能够提供更大的梯度，特别是在模型预测不准确时，能够促使更大的更新步长，从而加快收敛速度。
- 交叉熵损失函数:

    - 交叉熵损失函数的梯度不随预测概率的缩小而急剧减小。即使预测值与目标值相近，梯度也相对较大，从而确保在整个训练过程中有较快的收敛速度。
    - 交叉熵损失包括对数运算，当预测概率接近0或1时，对应的梯度会较大，这使得网络能够更迅速地调整预测，使其更接近目标值。
    - 交叉熵损失函数直接作用于概率分布上，它能够有效地衡量两个概率分布之间的差异，特别适用于分类问题中的 softmax 输出。

数据分布变化

问题：促销活动期间，用户行为和搜索需求发生了显著变化，导致训练数据和实时数据分布不一致。模型对新数据的泛化能力不足，导致排序效果下降。

- 特征工程和数据预处理
    - 增加时间特征: 例如是否在促销期间、促销天数等
    - 构建交叉特征：是否在促销期间购买
- 数据增强和采样策略
    - 数据增强：在模型训练过程中，增加促销活动期间的数据样本，使模型能够更好地学习这种特殊时期的用户行为模式。
    - 时间加权采样: 对于不同时期的数据样本，赋予不同的权重，使模型更关注近期的数据。
- 模型更新和在线学习
    - 促销期间，频繁更新模型，确保模型能够及时适应数据分布的变化。
    - 在线学习: 采用在线学习方法，使模型能够动态调整和更新参数，适应新的数据分布。
- 监控：及时发现数据发生偏移

增量在线训练模型有什么需要注意的地方

- **稳定性与保守性**
    - 确保增量更新不会引入过大的波动，避免模型的不稳定性。

    - 在模型参数更新过程中，可以采用一些控制策略，如学习率的调整或者增量更新的幅度限制，以保证模型参数的平稳更新。
- **设计增量更新策略**：确定模型参数的哪些部分需要增量更新，如权重、偏置等。
- 数据处理一致性、实时性
- 注意样本均衡
- **增量训练算法选择**：选择合适的增量训练算法，如增量梯度下降法、在线学习算法等。
- **评估与优化**：定期评估增量训练效果，根据评估结果调整增量训练的策略和参数。

视频广告预估，应该怎样做

- 分类任务：分类任务的核心思想是将广告时长划分为几个离散的时间段（如0-5秒、5-10秒、10-20秒等），然后预测视频广告的观看时长属于哪个时间段。适用于需要将广告时长映射到特定的时间段，如用户有多种行为路径时，或者时长需要作为触发其他业务逻辑的条件。
    - 特征工程:
        - 用户特征: 用户的基本属性、历史观看行为、兴趣标签等。
        - 广告特征: 视频长度、广告内容类型、广告历史点击率等。
        - 上下文特征: 广告播放时间、设备类型、网络环境等。
    - 模型
        - **常见分类模型**: 逻辑回归、决策树、随机森林、XGBoost、LightGBM等。
        - **深度学习模型**: 使用多层感知器（MLP）或卷积神经网络（CNN）结合用户特征和广告内容进行分类。
    - loss：交叉熵损失
    - **评估指标**: 准确率（Accuracy）、宏观平均F1分数（Macro F1）、AUC等。

- 回归任务：回归任务的目标是直接预测用户观看广告的时长，这是一个连续值的回归问题。，适用于需要精确预估广告时长的场景，如广告定价策略、广告效果优化等。
  - 直接使用广告的实际观看时长作为标签。
  - **特征工程**:
    - **用户特征**: 用户画像、历史广告交互行为、用户兴趣标签等。
    - **广告特征**: 视频内容、广告时长、广告质量评分等。
    - **上下文特征**: 时间、地点、设备、网络环境等。
  - **模型选择**
    - **回归模型**: 线性回归、决策树回归、随机森林回归、XGBoost回归、LightGBM回归等。
    - **深度学习模型**: MLP、RNN、LSTM、Transformer等，结合用户行为序列的时间依赖性进行建模。
  - **模型训练与评估**
    - **损失函数**: 使用均方误差（MSE）或均方根误差（RMSE）进行优化。
    - **评估指标**: R-squared、MAE、RMSE等。
    - **处理异常值**: 需要对长尾分布或异常值进行处理，如对时长取对数或采用鲁棒回归技术
- 上下游目标一致性
  - 在设计和优化视频广告时长预估模型时，需要确保预测时长与广告主和平台的目标一致。例如，如果广告的目标是提升品牌曝光度而非直接转化，那么在训练过程中可能需要调整样本的权重或修改损失函数，使得模型更关注那些能够带来较长观看时长的广告。
- 总结
  - **分类任务**: 适合对广告时长进行区间划分，适用于需要触发不同逻辑的场景。
  - **回归任务**: 适合需要精确预测广告观看时长的场景，通常可以提供更细粒度的预估。
  - **上下游一致性**: 确保模型优化方向与实际业务目标一致，避免模型效果偏离业务需求。

在一次重要的促销活动中，用户行为模式出现点击率上升但转化率下降的问题，可能涉及多个方面的原因。以下是一些可能的解决方案：

- 商品侧：
  - 商品质量、好评差评、复购率、历史加购率
  - 价格变化
  - 信息展示
- 用户侧：
  - 个性化：用户购买意图，历史是否购买过改商品、品类
  - 用户购买力分析
- 排序侧调整：
  - 优先展示转化高的商品

deepwalk

- 基本概念：DeepWalk的图构建主要围绕在一个图中生成随机游走路径，然后将这些路径转化为“句子”形式，用类似自然语言处理的方法来训练图节点的嵌入。通过这种方法，DeepWalk能够在图中自动学习到节点的特征表示，使得在嵌入空间中相似的节点相互靠近。

怎样构建图

- **定义图结构**：DeepWalk的输入是一张图（Graph），可以是有向图或无向图，带权图或不带权图。图由节点（nodes）和边（edges）组成，边连接节点，表示节点之间的关系。
- **图的表示形式**：通常图以邻接表（adjacency list）或邻接矩阵（adjacency matrix）的形式表示，具体选择取决于图的稀疏性和规模。在DeepWalk中，使用邻接表可以有效地进行随机游走操作。

- 基于商品相似度的商品邻接表：基于内容的相似度，如果两个商品在特征（如类别、品牌、价格、颜色、材质等）上相似，它们之间可以建立一条边。
- 基于协同过滤的相似度：
  - 基于用户-商品交互矩阵，计算商品之间的协同过滤相似度（如皮尔逊相关系数、余弦相似度等），建立商品之间的边。此过程类似于基于内容的相似度，只是数据来源和相似度计算方法不同。
- 共同加购：如果两个商品在用户的加购历史中经常被一起加入购物车，它们之间可以建立一条边。
  - 统计所有用户的订单数据。
  - 对于每一个用户，遍历其订单中的商品对，增加这些商品对之间的连接次数。
  - 使用商品对之间的购买次数作为边的权重，构建邻接表。

训练序列生成

- 随机游走Random Walks
  - **初始化**：对每个节点，初始化一次随机游走（或多次）。
  - **随机游走过程**：从当前节点开始，每一步随机选择一个相邻节点作为下一个访问节点，重复这个过程，直到达到预设的步长（walk length）。
- 生成上下文（Context Generation）
  - 对于每个随机游走生成的序列，将其看作一个句子，节点看作词。
  - 使用滑动窗口（window size）在每个随机游走序列上确定中心节点和其上下文节点（类似于词的上下文），以生成训练样本。动窗口大小为2：
- **训练节点嵌入（Node Embedding Training）**：
  - 使用类似于 Word2Vec 的 Skip-Gram 模型来训练节点嵌入。Skip-Gram模型尝试最大化给定节点预测其上下文节点的概率。
  - 损失函数（Loss Function）通常是负采样（Negative Sampling）技术的交叉熵损失，用于高效地优化。

Fp-growth

FP-Growth 提供了一种高效挖掘频繁商品组合的方法，根据用户行为序列构建用户-商品交互矩阵，然后构建商品相似度矩阵，优化了协同过滤的计算复杂度和推荐精度

优势

- **减少计算复杂度**：通过提前挖掘频繁项集，减少了在大规模数据上计算相似度的开销。
- **提高推荐精度**：频繁项集能够反映用户对商品组合的偏好，利用这些信息可以生成更符合用户兴趣的推荐。
- **增强模型可解释性**：频繁项集提供了商品之间的共现信息，有助于解释推荐结果。

实现流程

- **数据准备**：
  - 从用户历史行为（浏览、购买、加购等）中提取数据，构建用户-商品交互矩阵。

    ```
    用户 A: 牛奶, 面包, 黄油
    用户 B: 牛奶, 面包
    用户 C: 面包, 黄油
    用户 D: 牛奶, 黄油
    用户 E: 牛奶, 面包, 黄油, 果酱
    ```

  - 将用户的行为序列（如购买过的商品列表）作为事务集输入 FP-Growth 算法。
- 频繁项集
  - 支持度：出现次数n的才被记录
  - 根据 FP-Growth 找到的频繁项集，只在频繁出现的商品之间计算相似度，减少计算不必要的商品对。

频繁项集	支持度
{牛奶}	4
{面包}	4
{黄油}	4
{牛奶, 面包}	3
{牛奶, 黄油}	3
{面包, 黄油}	3
{牛奶, 面包, 黄油}	2

- 统计频繁项集出现的次数，使用Jaccard Similarity计算相似度

$$JaccardSimilarity = \frac{\text{共现次数}}{\text{两个商品各自出现次数之和} - \text{共现次数}}$$

$$JaccardSimilarity(\text{牛奶，面包}) = \frac{3}{4+4-3} = 0.6$$

  - 在推荐时，通过频繁项集的结果，可以更快速地定位推荐候选商品，减少在线推荐的延迟。

问题

- 进去是负责什么内容，业务场景
- 整体的一个业务流程是什么样子，比如一次业务迭代，数据、算法、上线、特征服务这些
- 人员架构
- 对于广告，因为新来的一个广告肯定是无样本，一开始的流量调控是怎么做的，怎么去分发广告，丰富样本，在有一定量的样本后，怎么去做召回和排序
- 冷启动在广告中是不是比较重要

一个新来的广告怎样处理

在广告系统中处理新广告（cold-start ads）的场景是一个典型的冷启动问题。对于新上线的广告，由于缺乏历史数据和用户交互信息，我们需要采用特定的策略来控制流量分发、丰富样本、以及后续的召回和排序。下面是一个常见的流程：

初期流量调控和样本丰富策略

对于新广告，初期需要通过一定的策略进行流量调控来丰富样本数据。这通常包括以下几步：

A. 冷启动阶段流量分配策略

- **小流量曝光**: 在广告刚上线时，将其放入一个冷启动广告池中，给它分配少量的流量。这种小流量分配通常是随机或基于一些简单规则（如平均曝光）的，目的是获取初始的用户行为数据（如点击、停留时间等）。
- **控制曝光频率**: 设定广告的初始曝光频率，确保它能被展示给足够多的用户但又不会过度曝光，避免浪费资源。
- **用户分组策略**: 可以选择对不同用户群体（如新用户、活跃用户、回流用户等）进行均匀的广告展示，这样可以快速测试广告在不同用户群体中的表现。
- **探索与开发（Exploration-Exploitation）策略**: 在新广告刚上线时，通过增加探索比例来展示新广告，逐渐收集用户反馈数据。在有足够样本后，逐渐转向开发，减少新广告的随机展示。

B. 冷启动模型和特征工程

- **基于内容的推荐**: 使用广告的内容特征（如标题、图片、描述等）进行初始推荐。通过分析广告的语义信息和视觉特征，找到与历史上表现好的广告相似的广告，从而推断它可能的用户喜好。
- **使用广告主信息**: 利用广告主的历史广告表现（如果有的话），包括广告主在其他广告上的点击率（CTR）、转化率（CVR）等作为初始特征。
- **上下文信息的利用**: 使用上下文特征，如时间、位置、用户设备等，结合内容信息进行初期流量的精准调控。

C. 激励机制

- **降低出价门槛**: 为新广告提供优惠或降低起始出价的门槛，以确保新广告在竞价中有一定的竞争力，从而获得初始曝光。

- **人工优选流量池**: 人为设置一部分流量池专门用于新广告的曝光。这部分流量池可能是点击率较高的用户群体，帮助快速获取点击数据。

2. 样本积累后的召回和排序

一旦新广告积累了一定量的样本数据（如足够的点击和转化数据），广告系统可以进入正常的召回和排序阶段。

A. 召回策略

- **基于用户兴趣召回**: 使用用户的历史行为（如点击、浏览、停留时间等）以及广告的特征来进行召回。利用用户画像信息和广告标签的匹配度来快速找到潜在的目标用户。
- **相似广告召回**: 通过向量化的方式将广告表示为向量，并与历史表现较好的广告进行相似度计算，找到与新广告相似的高质量广告，推测潜在的用户群体。
- **协同过滤召回**: 基于用户-广告交互矩阵，通过矩阵分解或图嵌入等方法，找到与用户兴趣相似的其他用户所点击的广告，并用作候选。

B. 排序策略

在召回阶段完成后，进入广告排序阶段，通常使用的模型有以下几种：

- **点击率预估模型（CTR Prediction Model）**：使用历史数据训练的点击率预估模型，根据用户-广告对的特征，预测广告被点击的概率。
- **转化率预估模型（CVR Prediction Model）**：预测用户点击广告后的转化概率。这类模型通常会结合点击数据和转化数据，以提升对最终目标的优化效果。
- **多目标排序模型（Multi-objective Ranking Model）**：除了 CTR 和 CVR 之外，还可能考虑其他目标（如用户留存率、广告主预算控制等）。通过多任务学习或强化学习的方式，综合优化多个目标。
- **深度兴趣网络（DIN）和深度兴趣进化网络（DIEN）**：利用用户的历史行为序列，对用户的即时兴趣和长期兴趣进行建模，以提升广告的个性化排序效果。

C. 模型调整与在线学习

- **实时调整**: 广告系统通常具备在线学习的能力，实时调整模型参数以适应新广告的动态变化。这样能够快速响应广告效果的变化和用户行为的变化。
- **重新训练模型**: 根据收集的新广告样本数据，定期对模型进行再训练，特别是增加与新广告相关的特征，以持续提升模型的准确性。

总结

通过以上步骤，广告系统可以有效地解决新广告的冷启动问题。在冷启动阶段，通过合理的流量分发策略和冷启动模型快速积累样本；在积累足够样本后，使用成熟的召回和排序策略，确保广告获得较好的展示和点击效果。整个流程需要不断地监控和优化，以适应广告主和用户的动态变化。

汤普森采样在冷启动中的应用

- 是一种在多臂老虎机（Multi-Armed Bandit）问题中非常有效的探索-开发（Exploration-Exploitation）策略，也被应用于广告投放、推荐系统等需要进行决策优化的场景中。它通过概率性地选择最优的选项，能够有效地平衡探索和开发。在电商平台中，汤普森采样可以用于冷启动策略，给曝光较少但加购率（或其他关键指标）高的商品更多的曝光机会。对于电商平台的商品推荐，特别是冷启动的场景，汤普森采样的核心思想是使用贝叶斯方法来估计每个商品的潜在收益（如点击率CTR或加购率CVR），并根据这些估计进行商品曝光决策。
- 实现
  - 数据准备
    - 筛选曝光少的商品
  - 定义商品的贝叶斯分布：在汤普森采样中，每个商品的加购率可以被认为服从一个**Beta分布**。Beta分布是贝叶斯统计中常用的先验分布，特别适合处理概率数据。
    - α_i：商品 i 被加购的次数（正向反馈）。

        - β_i : 商品 i 未被加购的次数（负向反馈）。
        - 初始时，对于一个新商品，没有历史数据，可以设定 都为1，这是一个非信息化先验，表示对其加购率的均匀分布假设。
    - 更新规则：
        - 如果商品被加购，更新规则为 $\alpha_i = \alpha_i + 1$。
        - 如果商品未被加购，更新规则为 $\beta_i = \beta_i + 1$。
    - 在每一轮决策中，根据商品的贝叶斯分布随机采样：
        - 从每个商品的 Beta 分布中采样一个加购率估计值 θ_i。使用 $np.random.beta(\alpha_i, \beta_i)$
        - 选择采样值最大的商品进行曝光。
        - 通过这种方式，曝光量大的商品加购率估计会逐渐趋于真实值，曝光少但加购率高的商品也有机会被选中。
- 汤普森采样的优势
    1. **平衡探索和开发**: 汤普森采样通过贝叶斯采样方法自动实现了探索（探索新商品）和开发（开发已有商品）的平衡，不需要手动设置探索比例。
    2. **适应性强**: 随着数据的增多，采样分布会逐渐收敛，系统可以更加精准地推荐商品，适应性较强。
    3. **提升整体效果**: 通过给潜在表现好的新商品更多的曝光机会，汤普森采样可以快速识别出高质量商品，提高整体平台的加购率和转化率。
    4. **处理冷启动问题**: 它有效地解决了冷启动问题，确保新商品有机会获得曝光，同时不会完全忽视已有商品的开发。

为什么没用协同过率、直接使用模型迭代路线的原因

蒙特卡洛

蒙特卡洛方法的基本概念

1. **随机抽样**：蒙特卡洛方法通过随机抽样来生成一系列样本，这些样本用于逼近问题的解。由于使用了随机性，蒙特卡洛方法能够处理复杂的、多维的积分和概率问题。

2. **概率统计**：通过大量样本的统计特性来估计期望值、方差或其他统计量。随着样本数量的增加，估计的结果会逐渐趋于真实值（大数定律）。
3. **应用范围**：蒙特卡洛方法适用于以下场景：
    - 多维积分的数值求解（特别是在维数较高时，传统数值积分方法效率较低）。
    - 模拟复杂的物理系统（如分子模拟、气体动力学）。
    - 金融领域的风险评估和期权定价。
    - 人工智能中的蒙特卡洛树搜索（如围棋AI）。
4. 求π

```python
import random
def estimate_pi(num_samples):
    count_inside_circle = 0
    for _ in range(num_samples):
        x = random.uniform(-1, 1)
        y = random.uniform(-1, 1)
        if x**2 + y**2 <= 1:
            count_inside_circle += 1
    return 4 * count_inside_circle / num_samples
# Example usage
print(estimate_pi(1000000))
```