阿里粗排技术体系与最新进展

原创 王哲 DataFunTalk 2月21日

收录于话题

#原创精选 124 #阿里巴巴 23 #推荐算法 19 #大厂实践 40

阿里粗排技术体系与最新进展

00:00 31:55

分享嘉宾：王哲 阿里巴巴

编辑整理：乐远

语音朗读：蒋志新

出品平台：DataFunTalk

导读：在搜索、推荐、广告等需要进行大规模排序的场景，级联排序架构得到了非常广泛的应用。以阿里的在线广告系统为例，按顺序一般包含召回、粗排、精排、重排等模块。粗排在召回和精排之间，一般需要从上万个广告集合中选择出几百个符合后链路目标的候选广告，并送给后面的精排模块。粗排有很严格的时间要求，一般需要在10～20ms内完成打分。在如此巨大的打分量以及如此严格的RT需求

下，粗排是如何平衡算力、RT以及最后的打分效果呢？本文将为大家分享粗排技术体系以及粗排最新进展COLD，目前COLD排序系统已经在阿里定向广告各主要业务落地并取得了巨大的线上效果提升。

本文的分享，主要围绕下面几点展开：

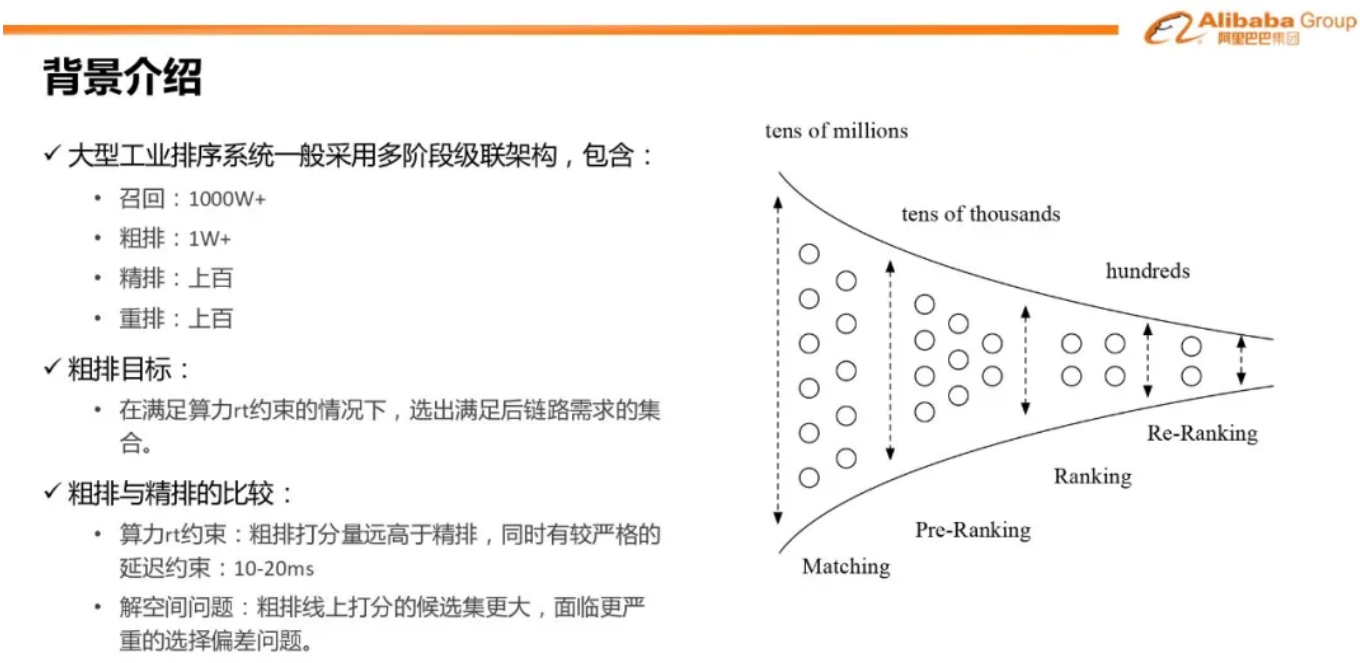
- 粗排的发展历史
- 粗排的最新进展COLD
- 粗排技术的总结与展望

01

粗排的发展历史

首先和大家分享下粗排的发展历史。

1. 粗排的背景



什么是粗排？一般的话，一个大型的工业级排序系统都会采用多阶段的排序架构，通常会包含四部分：召回、粗排、精排和重排。以阿里巴巴定向广告为例，召回的规模一般是千万左右，而粗排打分规模一

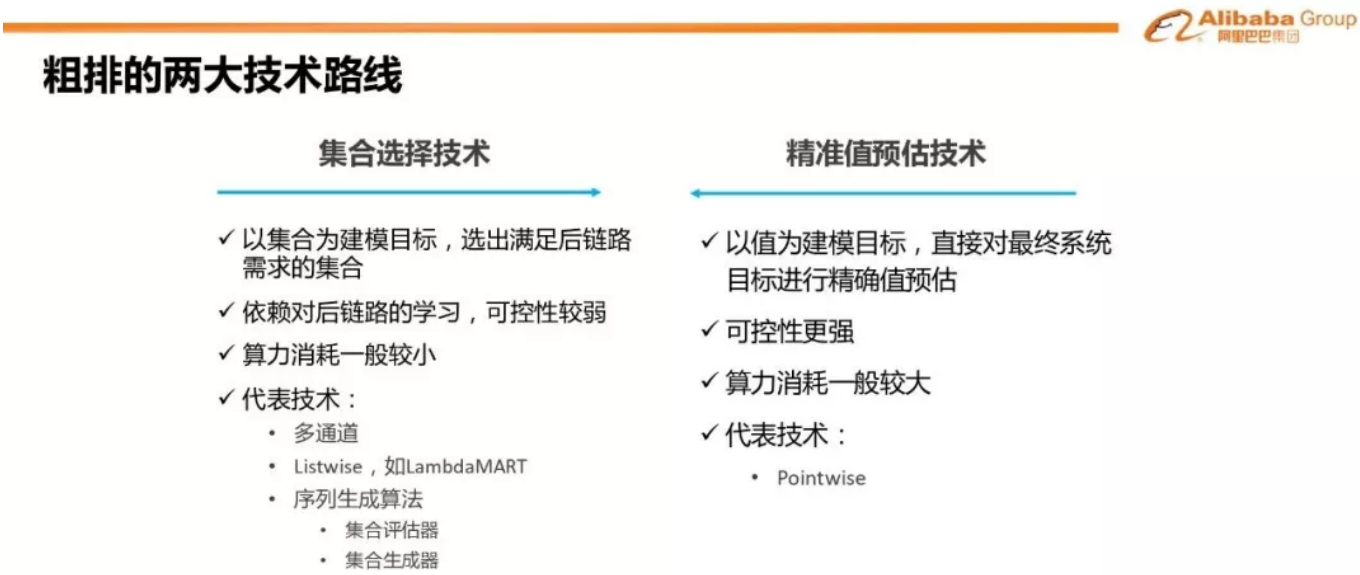
般是一万以上，精排和重排的规模一般是上百左右。粗排是处于召回和精排的一个中间阶段，目标是在满足算力RT约束的情况下，从上万个广告集合中选择满足后链路需求的候选广告集合。

粗排和精排有两点不同：

- 算力和RT的约束更严格：粗排的打分量远高于精排，同时有更严格的延迟约束，阿里巴巴定向广告的要求是10-20ms
- 解空间问题更严重：粗排和精排训练的时候使用的都是展现样本，但是线上打分环节粗排打分候选集更大，打分阶段距离展现环节更远，因此粗排阶段打分集合的分布和展现集合差距相比精排更大，解空间问题也更严重。

2. 粗排的两大技术路线

粗排的技术路线其实就是解决粗排一般该怎么迭代的问题，由于粗排是处于召回和精排之间的一个模块，因此粗排本身的迭代会受到前后链路的制约，因此需要站在整个链路的视角来看待这个问题。纵观整个链路，粗排一般存在两种技术路线：集合选择和精准值预估。



① 集合选择技术

集合选择技术是以集合为建模目标，选出满足后续链路需求的集合的方法，该技术路线在召回非常典型，这其实也非常符合粗排的定位。该方法优点是算力消耗一般比较少，缺点是依赖于对后链路的学

习，可控性较弱。

什么叫可控性？也就是说如果希望进行一些调整的话，由于这种方式依赖于通过数据回流对后链路进行学习，而数据回流往往比较慢，同时对数据量也有要求，可能需要后面链路的这些策略调整影响到比较大的流量之后，粗排才可以学习到，因此这种方式可控性比较弱，是偏被动的一种方式。

这种技术路线有以下常见的几种方法：

- 多通道方法：类似于召回，针对不同的目标构建不同的通道，然后分别选出不同的集合，然后再进行合并选择。

- Listwise方法：一般是直接建模集合的损失，典型算法如LamdaMART。为了更好理解listwise算法，这里提一下pointwise算法和pairwise算法，pointwise算法一般是点估计的方式，训练过程只考虑单条样本；而pairwise算法训练过程中会考虑当前样本和其它样本的相互关系，会构造这样的pair，并在训练的过程中引入这方面的pairwise loss，训练目标可能是正pair排在负pair的前面；Listwise更近一步，在训练的时候会考虑整个集合，会希望整个集合的指标如NDCG等达到最大化，如LamdaMART算法。

- 序列生成方法：直接做集合选择。一般包含集合评估器和集合生成器，算法过程如下：首先，用评估器对所有的item进行打分并选择一个得分最高的，作为集合中的第一个商品。接下来，再挑选第二个商品，把第一个商品和所有可能的第二个商品进行组合，并用评估器进行打分。之后，选择得分最高的集合，并持续使用类似于贪心的方式不断的搜索，最终得到一个最优的集合。

② 精准值预估技术

精准值预估技术直接对最终系统目标进行精确值预估，其实也就是pointwise的方式。

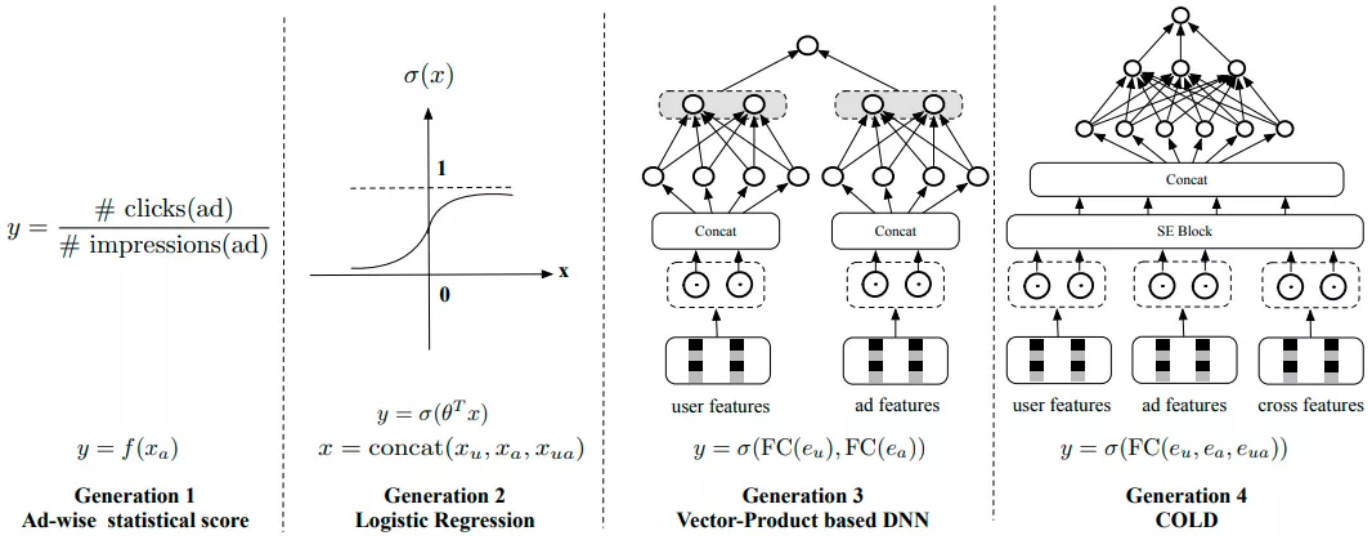
以广告系统为例，建模的目标一般是ECPM，即

$$ECPM=pCTR*bid$$

利用预估技术预估pCTR，然后预估bid，最终根据ECPM来进行排序，在粗排的话就是取排序的分最高的topK作为最终的集合。这种方式的优点是可控性强，因为是直接对整个目标进行建模，如果建模方

式做了调整的话，可以直接调整排序公式，调整预估模型，对整个链路的掌控力更强。缺点就是算力消耗比较大，而且预估越准确，算力消耗也越大。

3. 粗排的技术发展历史



粗排在工业界的发展历程可以分成下面几个阶段：

① 最早期的第一代粗排是静态质量分，一般是统计广告的历史平均CTR，只使用了广告侧的信息，表达能力有限，但是更新上可以做到很快。

② 第二代粗排是以LR为代表的早期机器学习模型，模型结构比较简单，有一定的个性化表达能力，可以在线更新和服务。

其中①②可以合称为“粗排的前深度学习时代（2016年以前）”。

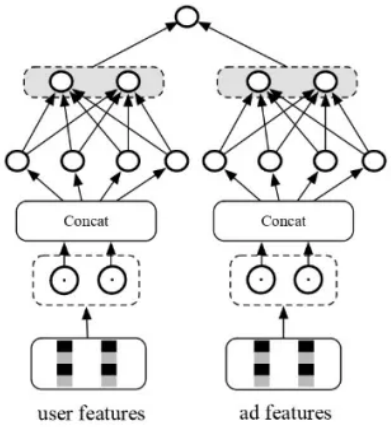
③ 当前应用最广泛的第三代粗排模型，是基于向量内积的深度模型。一般为双塔结构，两侧分别输入用户特征和广告特征，经过深度网络计算后，分别产出用户向量和广告向量，再通过内积等运算计算得到排序分数：

$$p = \sigma(v_u^T v_a)$$

③ 称为“粗排的深度时代-向量内积模型（2016）”。

粗排的深度时代-向量内积模型（2016）

- ✓ 双塔结构，两侧分别输入user特征和ad特征，经过DNN变幻后分别产出user向量和ad向量
- ✓ user侧网络可以引入transformer等复杂结构对用户行为序列进行建模
- ✓ 优点：
  - 内积计算简单，节省线上打分算力
  - user向量和ad向量离线计算产出，因此可以做的非常复杂而不用担心rt问题



$$y = \sigma(\mathrm{FC}(e_u), \mathrm{FC}(e_a))$$

Generation 3
Vector-Product based DNN

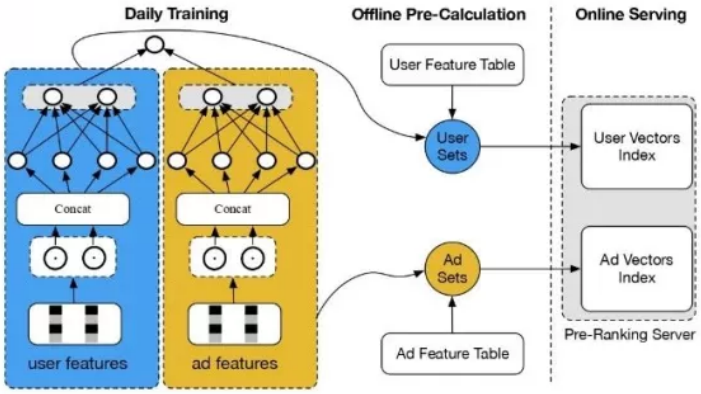
Covington P, Adams J, Sargin E. Deep Neural Networks for YouTube Recommendations. RecSys. 2016.

向量内积模型相比之前的粗排模型，表达能力有了很显著的提升，其优点：

- 内积计算简单，节省线上打分算力
- User向量和Ad向量离线计算产出，因此可以做的非常复杂而不用担心在线RT问题
- 双塔结构的user侧网络可以引入transformer等复杂结构对用户行为序列进行建模

向量内积模型的问题

- ✓ 模型表达能力受限
  - 难以很好的利用交叉特征
- ✓ 实时性较差
  - user向量和item向量一般需要提前计算好，这种提前计算会拖慢系统更新速度，难以对数据分布快速变化做出响应，例如双十一
  - 冷启动问题，对新广告不友好
- ✓ 迭代效率
  - user向量和item向量的版本同步影响迭代效率



然而仍然有许多问题：

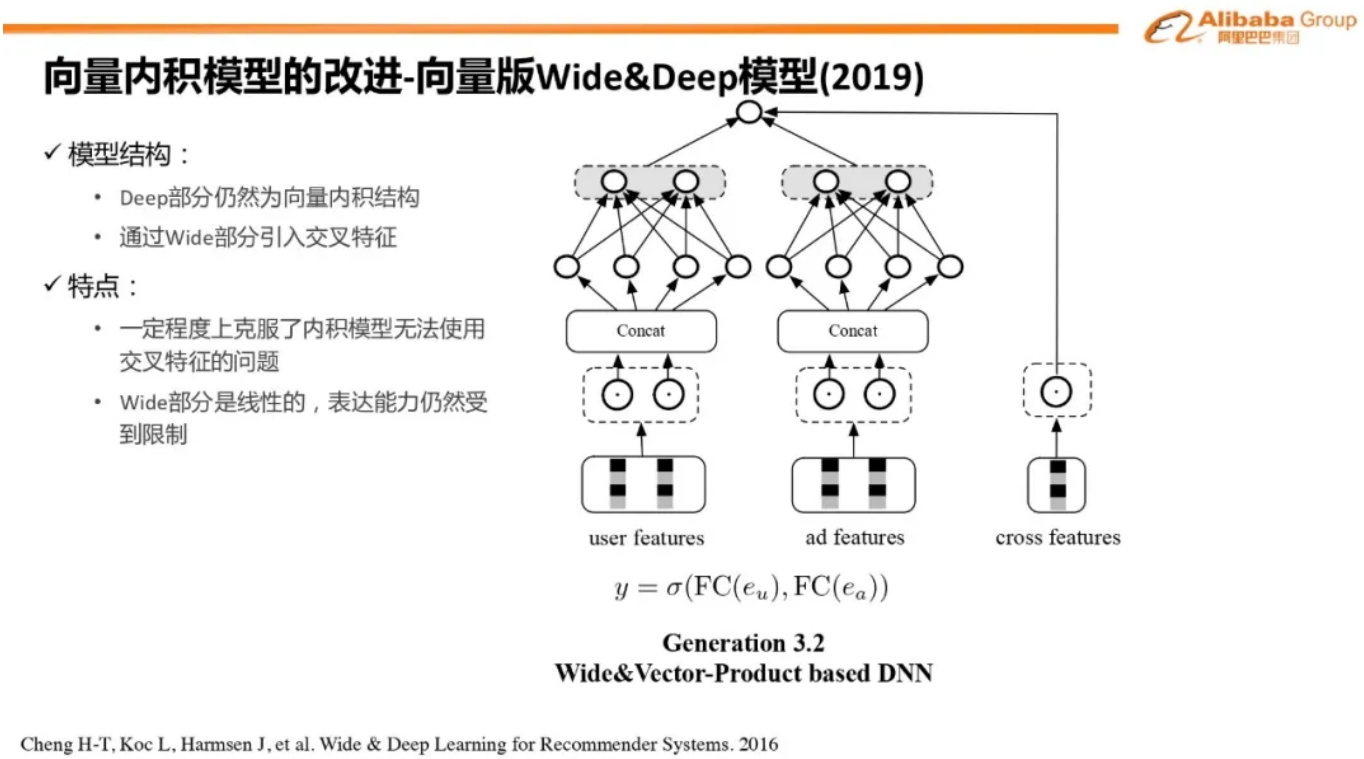
- 模型表达能力仍然受限：向量内积虽然极大的提升了运算速度，节省了算力，但是也导致了模型无法使用交叉特征，能力受到极大限制。

- 模型实时性较差：因为用户向量和广告向量一般需要提前计算好，而这种提前计算的时间会拖慢整个系统的更新速度，导致系统难以对数据分布的快速变化做出及时响应，这个问题在双十一等场景尤为明显。

- 存在冷启动问题，对新广告、新用户不友好

- 迭代效率：user向量和item向量的版本同步影响迭代效率。因为每次迭代一个新版本的模型，分别要把相应user和item向量产出，其本身迭代流程就非常长，尤其是对于一个比较大型的系统来说，如果把user和item都做到了上亿的这种级别的话，可能需要一天才能把这些产出更新到线上，这种迭代效率很低。

针对向量内积模型的问题，也有很多相关的改进，典型的如下面这个方法。

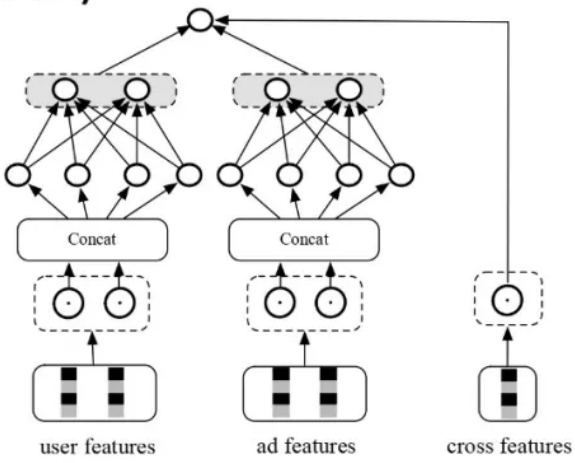


Cheng H-T, Koc L, Harmsen J, et al. Wide & Deep Learning for Recommender Systems. 2016

向量版Wide&Deep模型，deep部分仍然是向量内积结构，wide部分引入基于人工先验构造的user和ad的交叉特征，一定程度上克服了向量内积模型无法使用交叉特征的问题。然而该方法仍然有一些问题，wide部分是线性的，受限于RT的约束，不能做的过于复杂，因此表达能力仍然受限。

向量内积模型的改进-实时化(2019)

- ✓ User向量通过线上打分实时产出
- ✓ Ad向量仍然离线产出，但是更新频次加快
- ✓ 特点：
  - 通过实时打分，可以引入实时信息，实时性加强
  - 实时打分使向量内积模型的RT和算力优势减弱
  - 引入新的打分模型和ad向量版本一致性问题



$$y = \sigma(\mathrm{FC}(e_u), \mathrm{FC}(e_a))$$

Generation 3.2
Wide&Vector-Product based DNN

另外一个典型的改进方法是向量内积模型的实时化， user向量通过线上打分实时产出，ad向量仍然离线产出，但是更新频次增加。

通过实时打分，可以引入实时特征，实时性加强。然而实时打分使向量内积模型的RT和算力优势减弱，user模型处于RT的约束不能设计的很复杂，而且该方法还引入了新的打分模型和ad向量版本不一致的问题。

④第四代COLD，下一代粗排框架（2019）-算力感知的在线轻量级的深度粗排系统。下面将详细介绍该模型。

02

粗排的最新进展COLD

COLD：新一代粗排框架（2019）

✓ COLD : Computing power cost-aware Online and Lightweight Deep pre-ranking system

- 基于算法-系统Co-Design视角设计，算力作为一个算力与模型进行联合优化
- 模型结构没有限制，可以任意使用交叉特征
- 工程优化解决算力瓶颈
- 在系统实时系统，实时训练，实时打分，以应对线上分布快速变化

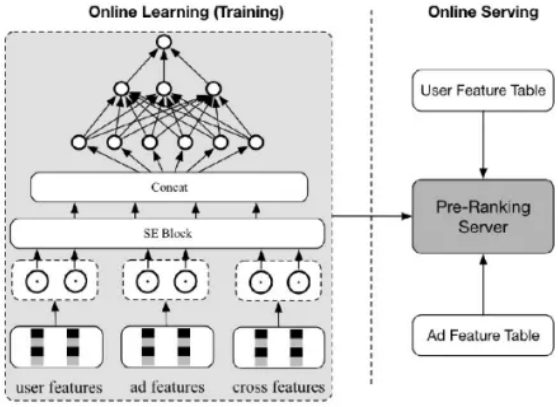


Figure 7: Infrastructure of fully online infrastructure of COLD pre-ranking system.

Wang, Zhe et al. COLD: Towards the Next Generation of Pre-Ranking System. DLP-KDD 2020

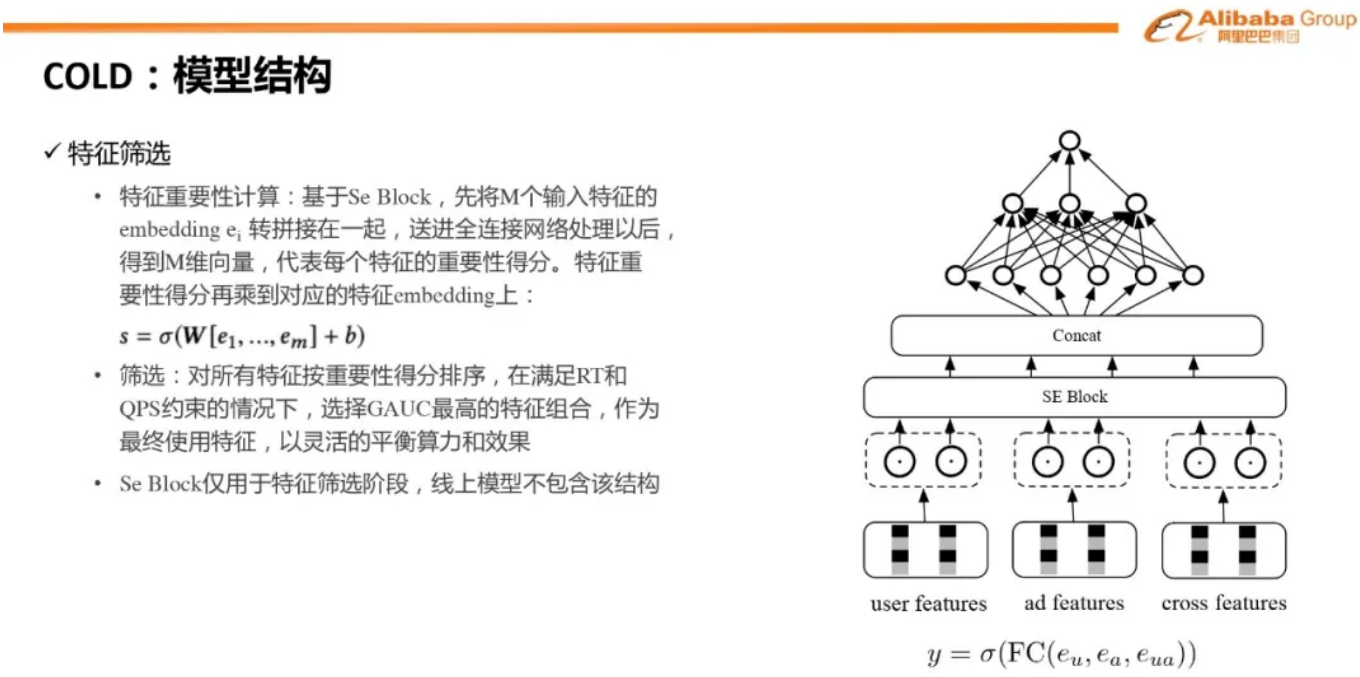
前面粗排的相关工作仅仅把算力看做系统的一个常量，模型和算力的优化是分离的。我们重新思考了模型和算力的关系，从两者联合设计优化的视角出发，提出了新一代的粗排架构COLD（Computing power cost-aware Online and Lightweight Deep pre-ranking system）。它可以灵活地对模型效果和算力进行平衡。COLD没有对模型结构进行限制，可以支持任意复杂的深度模型。这里我们把GwEN（group-wise embedding network）作为我们的初始模型结构。它以拼接好的特征embedding作为输入，后面是多层全连接网络，支持交叉特征。当然，如果特征和模型过于复杂，算力和延时都会难以接受。因此我们一方面设计了一个灵活的网络架构可以进行效果和算力的平衡。另一方面进行了很多工程上的优化以节省算力。

总结为以下几点：

- 基于算法-系统Co-Design视角设计，算力作为一个变量与模型进行联合优化
- 模型结构没有限制，可以任意使用交叉特征
- 工程优化解决算力瓶颈
- 在线实时系统，实时训练，实时打分，以应对线上分布快速变化

1. 模型结构

① 特征筛选



精简网络的方法有很多，例如网络剪枝（network pruning)、特征筛选（feature selection)、网络结构搜索（neural architecture search)等。我们选择了特征筛选以实现效果和算力的平衡，当然其他技术也可以进行尝试。具体来说，我们把SE (Squeeze-and-Excitation) block引入到了特征筛选过程中，它最初被用于计算机视觉领域以便对不同通道间的内部关系进行建模。这里我们用SE block来得到特征重要性分数。假设一共有M个特征，e_i表示第i个特征的embedding向量，SE block把e_i压缩成一个实数s_i。具体来说先将M个特征的embedding拼接在一起，经过全连接层并用sigmoid函数激活以后，得到M维的向量s：

$$s = \sigma(W[e_1, \ldots, e_m] + b)$$

这里向量s的第i维对应第i个特征的重要得分，然后再将s_i乘回到e_i，得到新的加权后的特征向量用于后续计算。

在得到特征的重要性得分之后，我们把所有特征按重要性选择最高的top K个特征作为候选特征，并基于GAUC、QPS和RT指标等离线指标，对效果和算力进行平衡，最终在满足QPS和RT要求情况下，选

择GAUC最高的一组特征组合，作为COLD最终使用的特征。后续的训练和线上打分都基于选择出来的特征组合。通过这种方式，可以灵活的进行效果和算力的平衡。

注意Se Block仅用于特征筛选阶段，线上模型不包含该结构。

② 基于scaling factor的结构化剪枝

此外COLD还会进行剪枝，做法是在每个神经元的输出后面乘上一个gamma，然后在训练的loss上对gamma进行稀疏惩罚，当某一神经元的gamma为0时，此时该神经元的输出为0，对此后的模型结构不再有任何影响，即视为该神经元被剪枝。

在训练时，会选用循环剪枝的方式，每隔t轮训练会对gamma为0的神经元进行mask，这样可以保证整个剪枝过程中模型的稀疏率是单调递减的。

$$\min_{w,\lambda} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, W, \gamma)) + R_s(\gamma)$$

这种剪枝方法在效果基本不变的情况下，粗排GPU的QPS提升20%。

最终模型是7层全连接网络。

2. 工程优化

为了给COLD使用更复杂的特征模型打开空间，工程上也进行了很多优化。在阿里定向广告系统中，粗排的线上打分主要包含两部分：特征计算和网络计算。特征计算部分主要负责从索引中拉取用户和广告的特征并且进行交叉特征的相关计算。而网络计算部分，会将特征转成embedding向量，并将它们拼接进行网络计算。

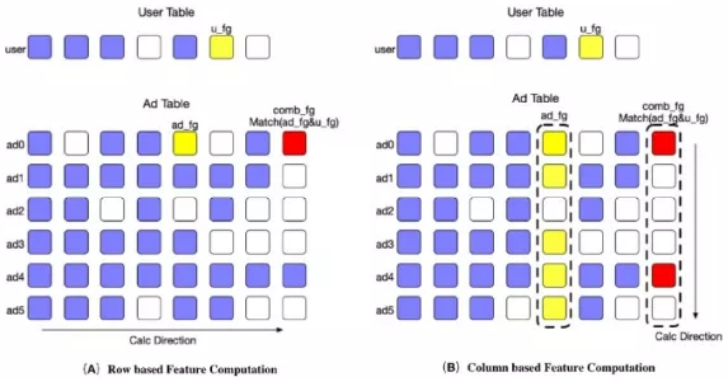
COLD：工程优化

✓ 并行优化：

- 将打分请求拆包以后，在特征计算和模型计算的各个地方，尽可能进行多线程优化

✓ 列计算转换:

- 行计算方式：逐个广告计算在不同feature group下的特征，存在访存不连续的，有冗余遍历，查找的问题。
- 列计算方式：因为同一个feature group的计算方法相同，因此可以按列进行特征计算，对同一列上的稀疏数据进行连续存储，之后利用MKL优化单特征计算，使用SIMD (Single Instruction Multiple Data)优化组合特征算子，以打到加速的目的。



① 并行化

为了实现低时延高吞吐的目标，并行计算是非常重要的。而粗排对于不同的广告的计算是相互独立的，因此可以将计算分成并行的多个请求以同时进行计算，并在最后进行结果合并。特征计算部分使用了多线程方式进一步加速，网络计算部分使用了GPU。

② 行列转换

特征计算的过程可以抽象看做两个稀疏矩阵的计算，一个是用户矩阵，另一个是广告矩阵。矩阵的行是batch_size，对于用户矩阵来说batch_size为1，对于广告矩阵来说batch_size为广告数，矩阵的列是featue group的数目。常规计算广告矩阵的方法是逐个广告计算在不同feature group下特征的结果，这个方法符合通常的计算习惯，组合特征实现也比较简单，但是这种计算方式是访存不连续的，有冗余遍历、查找的问题。事实上，因为同一个feature group的计算方法相同，因此可以利用这个特性，将行计算重构成列计算，对同一列上的稀疏数据进行连续存储，之后利用MKL优化单特征计算，使用SIMD (Single Instruction Multiple Data)优化组合特征算子，以达到加速的目的。

③ Float16加速

COLD：工程优化

✓ Float16加速：

- linear log trick

$$linear\_log(x) = \begin{cases} -log(-x) - 1 & x < -1 \\ x & -1 \le x \le 1. \\ log(x) + 1 & x > 1 \end{cases}$$

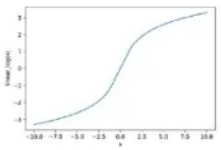


Figure 6: The linear_log function

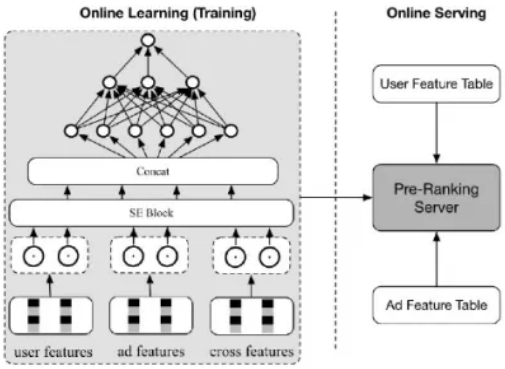


Figure 7: Infrastructure of fully online infrastructure of COLD pre-ranking system.

对于COLD来说，绝大部分网络计算都是矩阵乘法，而NVIDIA的Turning架构对Float16和Int8的矩阵乘法有额外的加速，因此引入Float16计算对提升性能非常必要 。但是Float16会损失计算精度，特别是在sum-pooling的情况下，数值有可能超出Float16的范围。为了解决这个问题，一种方式是使用BN。但是BN本身的参数范围也有可能超过Float16。因此只能使用混合精度的方式，对于BN层使用Float32，而后面的层使用Float16。另一种方式是使用参数无关的归一化方式，例如log函数。但是log函数不能处理负数，并且输入值接近0的时候会输出绝对值较大的数字。因此我们设计了一种分段平滑函数，我们叫做linear_log来解决这个问题。

从函数图像可以看出，linear_log函数可以将Float32的数值处理到一个比较合适的范围。所以如果我们将linear_log函数放到第一层，那么就可以保证网络的输入参数在一个比较小的范围内。具体实践上，linear_log函数对COLD模型的效果基本没有影响。使用Float16以后，CUDA kernel的运行性能有显著提升，同时kernel的启动时间成为了瓶颈。为了解决这个问题，我们使用了MPS (Multi-Process Service)来解决kernel启动的开销。Float16和MPS技术，可以带来接近2倍的QPS提升。另外，使用阿里自研的含光800NPU专有硬件，替代原来的GPU，QPS进一步提升约1倍。

3. 在线服务架构

COLD：在线服务架构

- ✓ 在线实时ODL训练
- ✓ 在线实时inference
- ✓ 更及时响应数据分布变化，对新广告更友好
- ✓ 实时架构对模型迭代和在线A/B测试都更有利

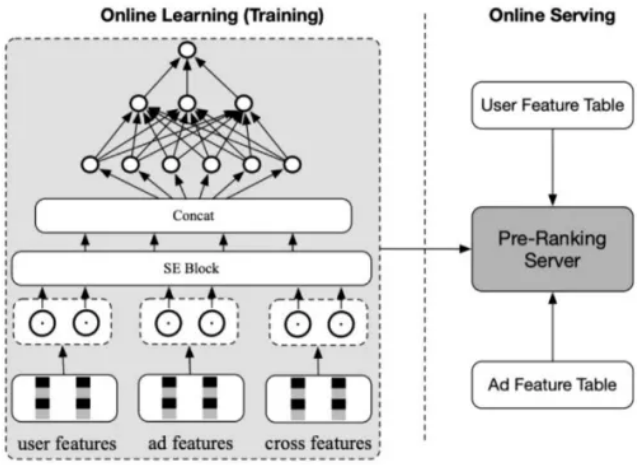


Figure 7: Infrastructure of fully online infrastructure of COLD pre-ranking system.

COLD没有限制模型的结构，训练和在线打分都是实时化的，可以带来以下两个优点：

- 在线学习的引入使COLD与向量内积模型相比，可以更及时的响应数据分布的变化，对新广告冷启动也更为友好。
- 实时架构对于模型迭代和在线A/B测试都更有利。向量内积模型由于用户向量和广告向量需要提前计算好，在线A/B测试也更为困难。实时架构也使COLD模型可以更快的更新，避免了向量内积模型的更新延迟问题。

4. 实验结果

这里COLD模型使用了7层全连接的网络结构。离线评估指标除了GAUC之外，还包含了top-k recall，用于评估粗排和精排的对齐程度。

$$recall = \frac{|\{\text{top k ad candidates}\} \cap \{\text{top m ad candidates}\}|}{|\{\text{top m ad candidates}\}|}.$$

其中top k候选集合和top m候选集合均为粗排的输入打分集合。top k集合是粗排选出的，而 top m集合是精排选出的，排序指标是eCPM(eCPM = pCTR*bid)。这里的精排模型是DIEN。我们使用QPS (Queries Per Seconds, which measures the throughput of themodel) 和 RT (return time, which measures the latency of model)来评估系统性能的影响。

① 模型效果评估

实验结果

✓ 离线实验

Method	GAUC	Recall
Vector-Product based DNN Model	0.6232	88%
COLD	0.6391	96%
DIEN	0.6511	100%

✓ 在线效果

Time	CTR lift	RPM lift
Normal Days	+6.1%	+6.5%
Double 11 Event	+9.1%	+10.8%

✓ 2019年以来，COLD已经在阿里妈妈定向广告各主要业务线落地，并取得了可观的线上效果提升。

离线效果评估可以看到COLD在GAUC和Recall上都优于向量内积模型。在线效果上，COLD与向量内积模型相比在日常CTR +6.1%,RPM + 6.5%。双十一CTR+9.1%，RPM+10.8%，提升显著。

② 系统性能评估

实验结果

✓ 不同模型结构的性能表现

Model	QPS	RT
Vector-Product based DNN Model	60000+	2ms
COLD	6700	9.3ms
DIEN	629	16.9ms

Model	QPS	RT	GAUC
COLD (No Cross Features)	6860	8.6ms	0.6281
COLD *	6700	9.3ms	0.6391
COLD (All Features)	2570	10.6ms	0.6467

从上面表格可以看到，向量内积模型的系统性能最好，而精排的DIEN的系统性能最差，COLD则在两者之间取得了平衡。

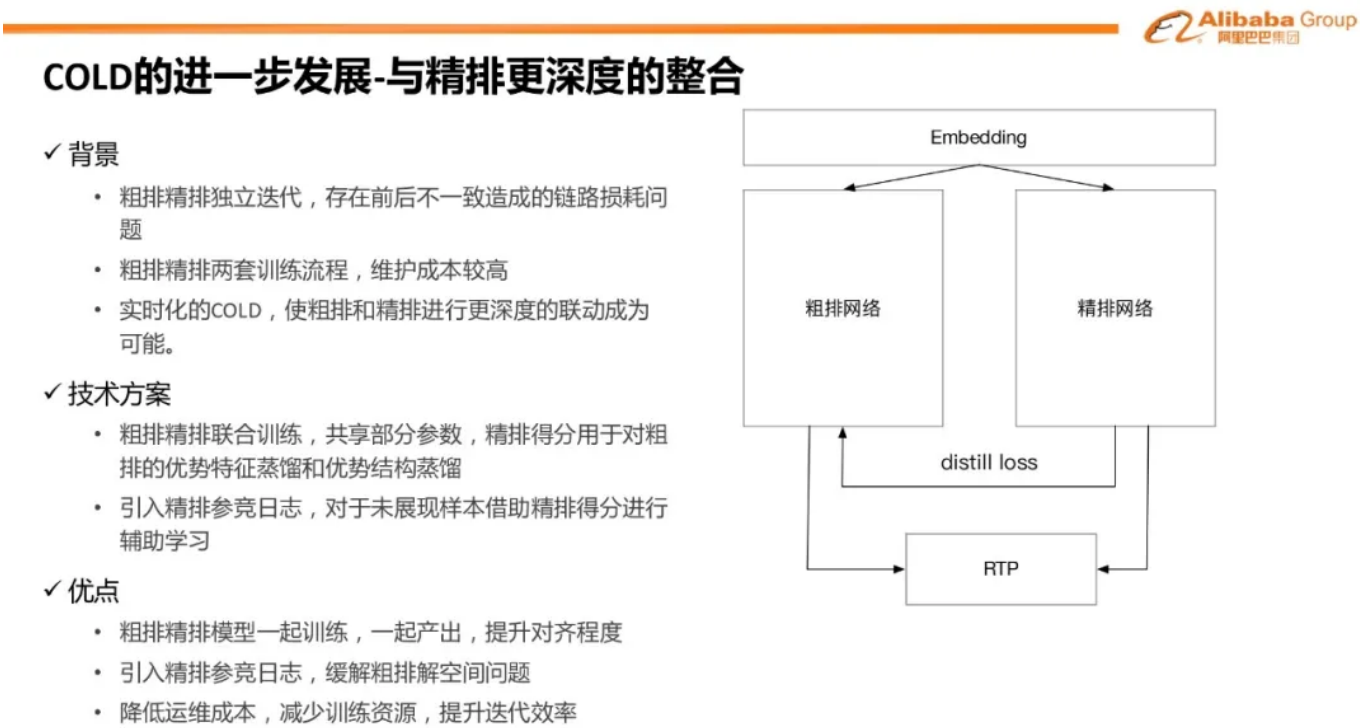
COLD在得到特征重要性分数以后，会选出不同的候选特征，并基于离线指标进行特征选择。第二个表列了几组供选择的特征，可以看到COLD是考虑效果和系统性能以后的折中。

Table 5: Comparison of system performance by computing with different GPU precisions

Precision	QPS
COLD (Float32)	2800
COLD (Float16)	3400
COLD (Float16+MPS)	6700

Table 5表明，工程优化上引入Float16和MPS优化以后，QPS提升了1倍，效果显著。

5. COLD的进一步发展



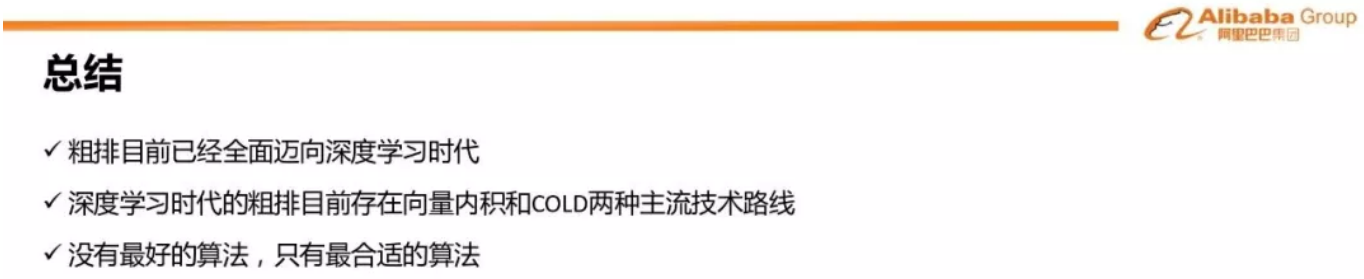
那么COLD的后面进一步怎么发展？答案是与精排进行更加深度的整合。目前粗排和精排都是各自独立迭代的，这样会造成一个前后不一致的问题，会带来一个链路的内耗。而且粗排和精排维护两套流程的话，本身维护成本也很高，尤其是像阿里巴巴定向广告这边，业务多而且复杂。另外，COLD和精排一样都是非常实时化的架构，这使得粗排和精排进行更深度的联动成为可能。

具体如何进行更加深度的整合呢？这里我们直接将粗排和精排的生产流程合二为一，联合训练，共享部分参数。另外也会借助特征蒸馏的方式，使用精排训练过程中的得分来指导粗排。在这个基础上，尝试引入精排参竞日志，对于未展现样本借助精排得分进行辅助学习。为什么引入精排参竞日志？其实前面提到过，因为粗排除了RT还有一个就是解空间的问题，粗排本身见到的集合比精排更多，但是它本身模型的能力却比精排更弱。所以为了缓解这个问题，引入精排参竞日志，这部分日志本身是没有label的，可以利用精排在这些label上进行打分，通过蒸馏的方式来引导粗排进行学习。为什么说这种方式在一定程度上能缓解解空间的问题？其实是建立在精排的模型能力比粗排更强，这个强很大程度体现在精排对交叉特征的利用上，在这种情况下，对于很多没有展现的user和item的pair对，精排相比于粗排其实是可以给出更精确的预估的，在这个前提下，借助精排更精确的预估分数，然后对粗排进行指导，在一定程度上是可以缓解解空间问题的。

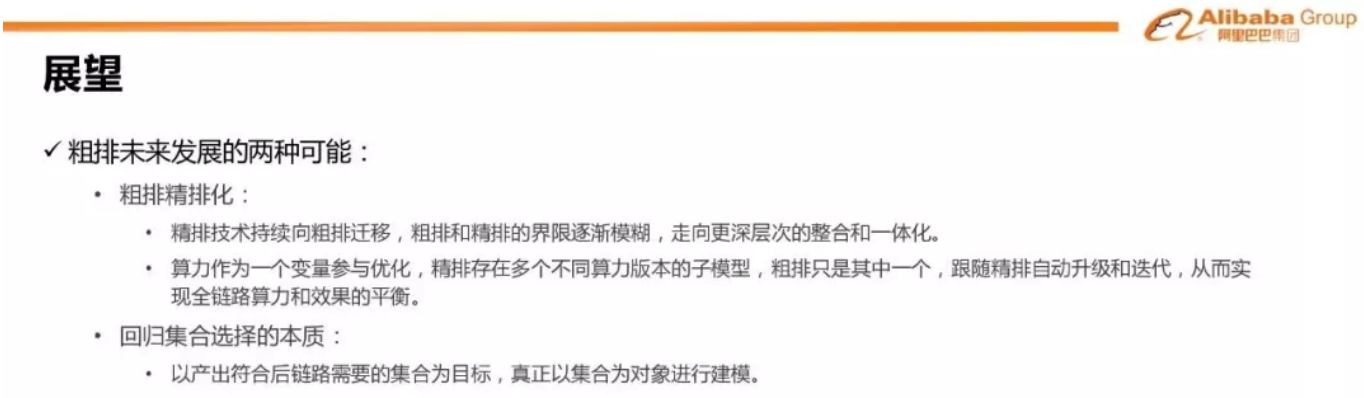
这种方案的好处是将粗排和精排的流程合二为一，粗排精排模型一起训练，一起产出，这样可以提升粗排和精排的对齐程度，解决前面提到的因为前后链路不一致造成的链路损耗问题。另外引入精排参竞日志，在一定程度上也能缓解解空间问题。而且还有一个好处就是降低运维成本，减少了训练资源，提升我们的迭代效率。

03

粗排技术的总结与展望



最后做一下总结，目前整个粗排基本已经全面迈向深度学习时代，而且深度学习时代的粗排主要存在向量内积和COLD两种主流技术路线。没有最好的算法，只有最合适的算法。不管是COLD还是向量内积，其实都有它适合的场景。有的团队在向量内积这条路线上持续迭代，其实也是受到本身算力RT的一些限制，包括本身工程团队等各方面因素的影响。而且很多团队的精排也还是向量内积结构，粗排就没有必要升级到COLD了。



最后，展望一下粗排会怎么发展。粗排未来会存在两种可能的路线。

一个是粗排精排化，也就是说精排技术持续向粗排迁移，粗排和精排的界限逐渐模糊，走向更深层次的整合和一体化。其实可以看到，粗排的迭代历程本身其实就是一个粗排精排化的过程，一般很多技术如向量内积，都是先在精排落地拿到效果后，再逐渐向粗排迁移。而且算力也会更深度的参与到整个模型的迭代过程中，未来可能的话甚至不会再分粗排和精排，精排本身可能会在引入算力作为一个约束的情况下，通过AutoML等方式，产出多个不同算力版本的子模型，粗排只是其中一个，跟随精排自动升级和迭代，从而实现全链路算力和效果的平衡。甚至算力足够大的话，可以不需要粗排了，直接用精排打分就可以了。

另外其实还存在另外一条路，就是回归集合选择的本质，以产出符合后链路需要的集合为目标，真正以集合为对象进行建模。这里需要清楚粗排的目标是什么？粗排的目标是选出后链路需要的集合。而目前粗排精排化的迭代思路其实是在做ranking，但是粗排其实只需要一个top k的集合，集合内部无序就可以了。而粗排精排化这种方式消耗了很大一部分算力在TopK的内部排序上，很多算力消耗在Top1和Top2怎么排好，从计算角度来看是一种资源浪费。当然这条技术路线还比较新，目前还在探索中。但是这个方向是一个很有希望的方向，因为它回归了粗排的本质，直接面向粗排的最终目标，因此在算力的利用率和最终的天花板上是更高的。

参考资料：

【1】Covington P, Adams J, Sargin E. Deep Neural Networks for YouTube Recommendations. RecSys. 2016.