[搜索系统算法解析-OAA]

粗排

粗排 是搜索系统中排序环节的第一层，主要目标是从大量候选文档中快速筛选出一小部分高潜相关的文档，为后续的精排提供输入。粗排阶段通常需要在较低的计算成本下实现高召回率和初步的排序能力，兼顾效率和性能。

粗排的目标不仅是通过高效的排序手段完成候选集的过滤，更重要的是为后续的精排提供一个更优质、更聚焦的输入，从而在整个搜索链路中起到承上启下的关键作用。

从打分目标和能力上看，精排更加注重头部排序效果，专注于少量最优结果的精细化排序。而粗排需要从召回的多样化候选集里挑选一个覆盖主需和次需的优质子集，为精排提供足够的搜索空间。所以，粗排需要关注的是腰部、长尾文档的质量，确保为精排提供的子集包含更多潜在优质候选文档，而非过度优化头部排序。

模型架构

相比于采用单塔交互模型的精排阶段，粗排阶段则采用双塔模型保以证召回质量的同时兼顾计算效率和扩展性。

双塔模型

双塔模型由 **查询塔（Query Tower）** 和 **文档塔（Doc Tower）** 组成：

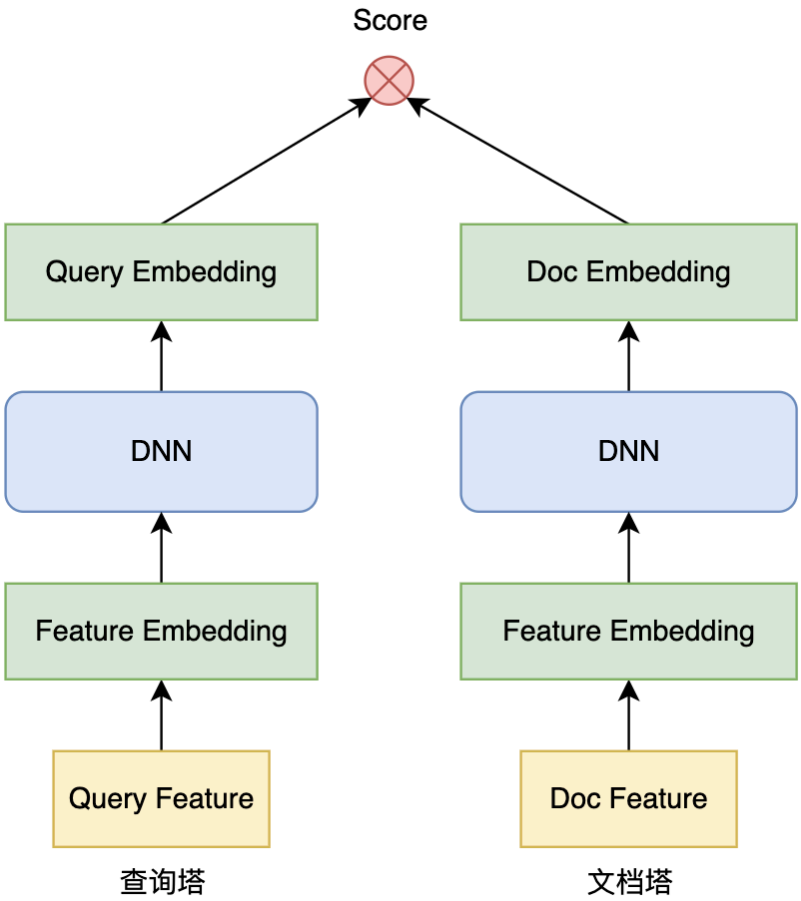
两个塔分别对Query和Doc进行独立的向量化表示，最后通过相似度计算（如内积或余弦相似度）评估二者的匹配度。

$$\text{Score}(q, d) = \text{Similarity}(f_q(q), f_d(d)) \tag{1}$$

- $f_q(q)$：Query信息的向量表示
- $f_d(d)$：Doc信息的向量表示
- Similarity：通常为内积或余弦相似度

当查询和文档映射到向量空间后，通过 **余弦相似度（Cosine Similarity）** 或 **内积（Dot Product）** 计算两者的相似度。模型训练时，优化目标是最大化相关查询-文档对的相似度，同时最小化不相关查询-文档对的相似度。

通常来说，使用 Cosine 比 内积 效果要好，Cosine 相似度通过归一化消除了向量长度的影响，即更加关注向量的 **方向**（角度），而内积则受向量长度的影响。另外，采用内积计算相似度时，对向量进行 **L2归一化** 等价于 Cosine。



查询塔

特征输入

查询塔的模型输入特征通常有三个维度：Query特征、User特征、User-Query交叉特征：

特征	类型
Query 文本	Query 特征
Query 属性特征（类目/NER/实体/主题/意图等）	
User 属性特征（User ID/性别/年龄/城市/画像等）	User 特征
User 历史搜索词/互动文档序列	

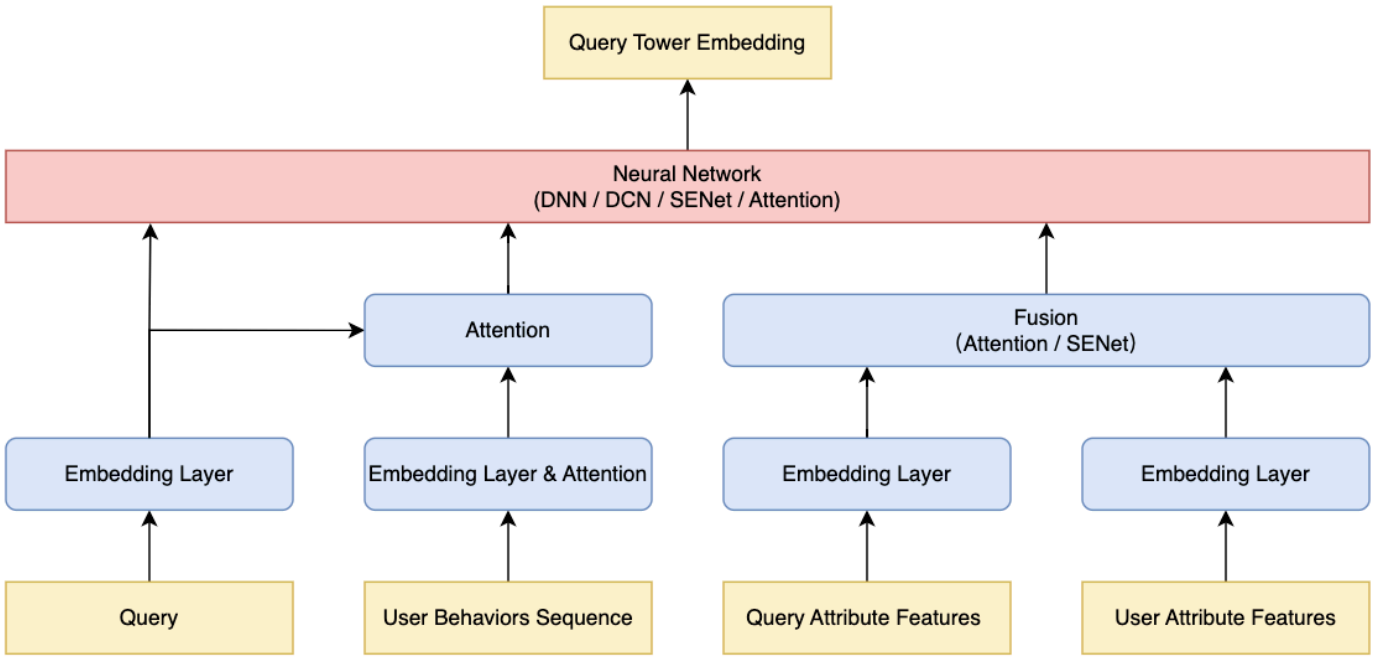
其中，

- Query文本 和 Query属性 中可以用文本表示的特征通常采用 **Sentence-BERT** 转化成向量表示（根据平台算力选择冻结BERT直接采用预训练向量，或随双塔模型同时训练BERT参数）
- 对于用户历史搜索词/点击文档序列，采用 **注意力机制（Attention）** 捕捉用户历史行为和查询词Query的相关性
    - 在序列中添加一个全零向量，模型在进行注意力计算时，可以选择将该零向量赋予最大注意力，可以避免在历史行为与当前查询完全不相关的场景中，模型强制关注到历史行为而带来的噪声
    - 对于不同时间周期的行为序列（实时、短期、长期）可以设计不同的模型策略

- User ID、离散化后的统计特征（分桶）等 **离散特征** 通过Embedding层进行编码，然后可以接入 **全连接层** 将输入特征的各个维度组合起来，并通过权重矩阵进行线性变换，使得模型可以捕捉到更复杂的特征关系
- User ID 特征可以建模关联到用户的历史行为、偏好以及其他相关信息，可以帮助模型做出 **个性化预测**
- User ID 特征具有高纬度和稀疏性，模型可能会学到不具有泛化能力的噪声而导致过拟合：
  - 使用 **哈希函数** 将ID空间映射到一个较小的哈希空间以减少内存使用
  - 使用 **特征降维** 技术（如主成分分析，PCA）对ID向量进行降维
  - 使用 **Dropout** 随机丢弃部分神经元以防止过拟合
- 对于缺少搜索消费行为的 User ID 特征的 **冷启动** 问题
  - **迁移学习：** 采用其他产品域的 User ID 向量
  - **预训练模型生成：** 利用预训练语言模型从用户的文本信息（如用户注册时填写的描述、行为日志、社交网络数据等）生成与用户相关的向量表示
  - **相似群体代替：** 查找相似且高活的 User ID Embedding 取平均

模型结构

根据模型输入的不同维度的特征以及其各自的属性，需要进行针对性的模型结构设计：



具体的：

- Query Text 通常采用 BERT 转化成向量表示（也可采用文本卷积神经网络替换）
- 用户行为序列可采用 Transformer 学习更好的向量表示（Transformer 内部的自注意力机制可以对序列内元素的相互关系进行有效的建模来实现更好的表达）
- 采用 注意力机制（Attention）来捕捉用户历史行为和查询词Query的相关性
  - 在序列中添加一个全零向量，模型在进行注意力计算时，可以选择将该零向量赋予最大注意力，可以避免在历史行为与当前查询完全不相关的场景中，模型强制关注到历史行为而带来的噪声
  - 对于不同时间周期的行为序列（实时、短期、长期）可以设计不同的模型策略

- Query/User 离散特征通过 Embedding层 进行编码，通常对编码后的多个离散向量进行拼接，将信息整合到一个向量中

在得到Query特征和User特征向量表示之后，需要经过神经网络实现特征间的交互，常用的网络结构有DCN、CIN、FiBiNet等，此外通常采用SENet对塔底层各通道的特征进行特征增强和过滤。

文档塔

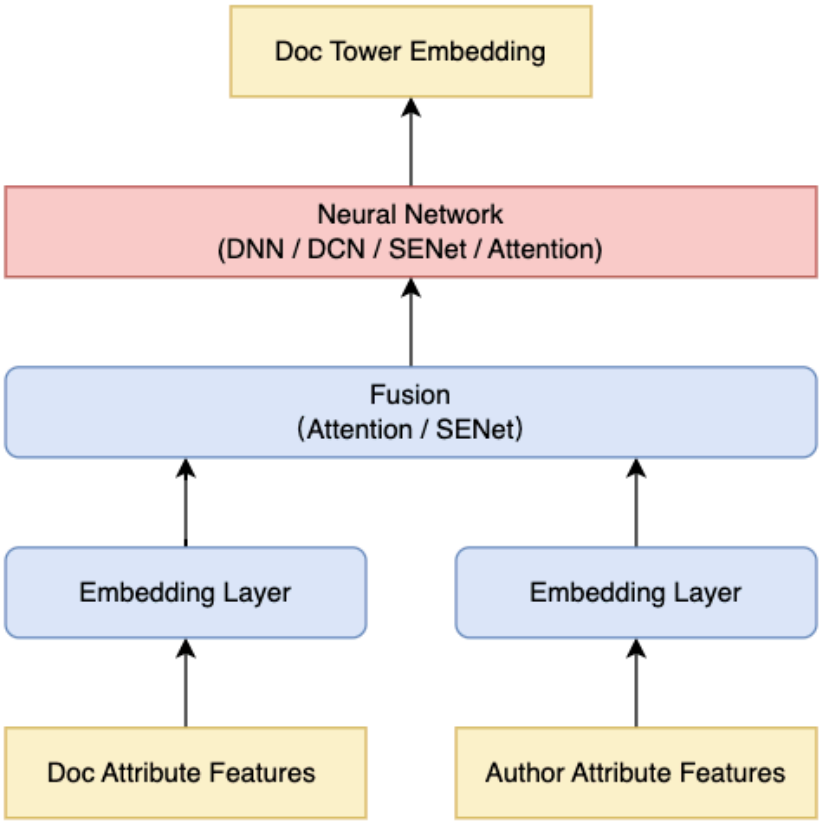
特征输入

查询塔的模型输入特征通常有三个维度：Doc特征（文档特征）、Author特征、Author-Doc交叉特征：

特征	类型
Doc 文本（标题/正文/OCR/ASR/评论等）	Doc 特征
Doc 多模态特征（图片/视频/音频等）	
Doc 属性特征（Doc ID/类目/标签/主题/点击 Query 等）	
Doc 互动特征（曝光/点击/点赞/收藏/评论等）	
Author 属性特征（Author ID/性别/年龄/城市/画像等）	Author 特征
Author 互动特征（发布/曝光/评论/粉丝等）	

模型结构

文档塔的模型结构可以设计为：

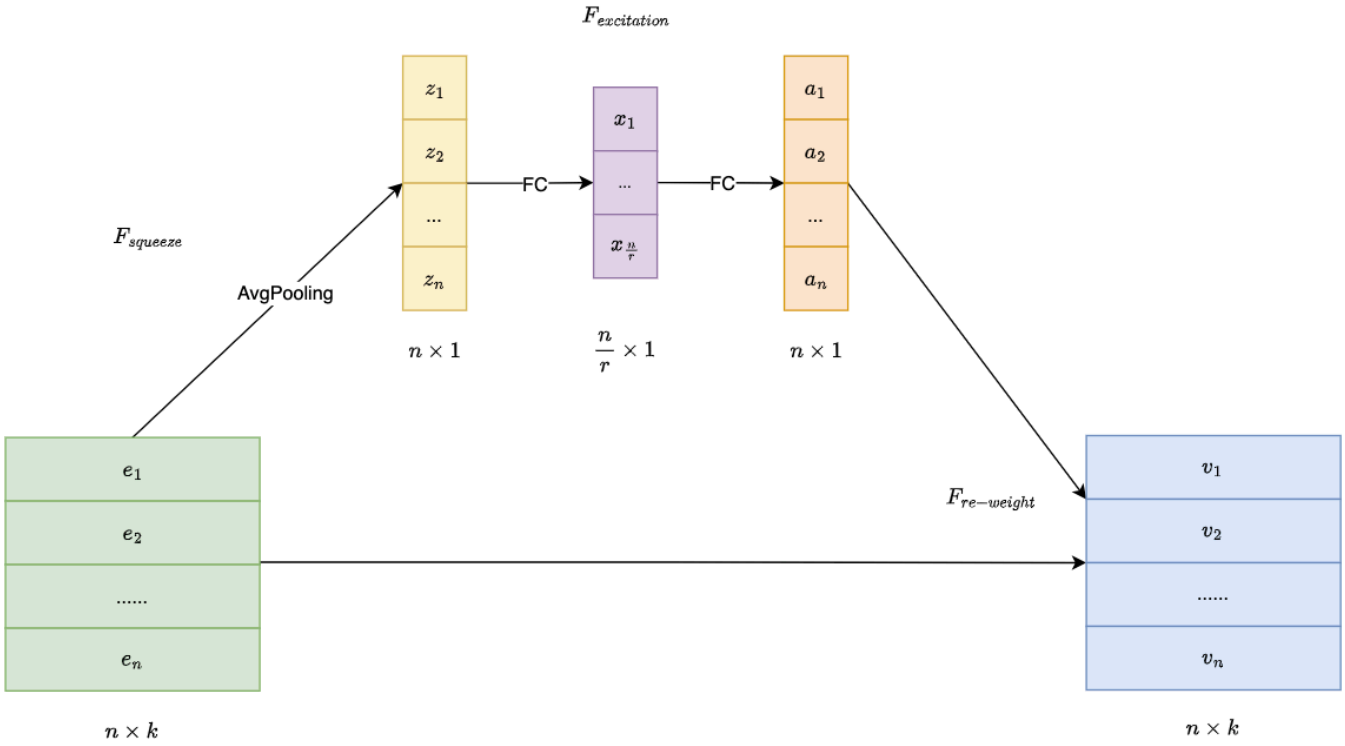


特征加权和特征交叉

传统的双塔中DNN结构一般由多个全连接层组成，每层通过加权和激活函数进行非线性变换，特征交互通过全连接层隐式学习特征交叉。在每一层，输入特征 $\mathbf{x}$ 和权重矩阵 $\mathbf{W}$ 的运算是标量相乘并求和的过程，在捕捉高阶交叉和复杂的特征关系时表达能力有限。如何更好的平衡显式和隐式的特征交叉，以及动态分配特征重要度是模型是提升模型性能的关键因素。下面是常见的特征加权和特征交叉网络。

SENet

SE-Net 引入 **Squeeze-and-Excitation (SE) Block** 来增强神经网络表示能力的架构。SE块通过自适应地调整各通道的权重，从而提高网络对有意义特征的关注能力。



- **Squeeze**：通过全局平均池化（Global Average Pooling）对每个通道的特征压缩为一个标量，生成一个通道描述符。即对每个通道进行信息整合，得到通道的全局特征表示
  - $$z_i = F_{sq}(e_i) = \frac{1}{k}\sum_{t=1}^{k} e_i^{(t)} \tag{2}$$
  - 其中有 n 个特征 e，每个特征可表示为k个向量（域，filed）
  - 该步骤将原始向量 $E = [e_1, \ldots, e_n]$ 挤压成统计向量 $Z = [z_1, \ldots, z_n]$，z_i 为标量
- **Excitation**：通道描述符通过一个小型的全连接网络（通常是一个两层的 MLP）进行进一步处理，生成每个通道的权重（或缩放因子）。即为每个通道分配一个重要性权重，增强重要通道，抑制不重要的通道
  - $$A = F_{ex}(Z) = \sigma_2(W_2\sigma_1(W_1 Z)) \tag{3}$$
  - Excitation采用两个全连接层（FC）学习权重，第一个FC层是降维层，参数为 W_1，降维的压缩比是超参数 r；第二个FC层通过参数 W_2 恢复维度

- σ_1 和 σ_2 为非线性激活函数，通常分别用 ReLU 和 Sigmoid
- 第一层通过压缩层减少维度，从而限制模型复杂度并帮助泛化，同时也使得模型聚焦于更有意义的通道依赖。采用 ReLU 引入非线性，学习通道之间的相互依赖性
- 第二层则将这种压缩后的信息恢复，并学习到每个通道的重要性。通过 Sigmoid 函数将输出映射到 [0,1] 的范围，为每个通道分配一个注意力系数。

- **Re-Weight**：对原始域向量 E 和域权重向量 A 进行 **逐域（field-wise）相乘**，输出新的向量 $V = [v_1, \ldots, v_n]$，**域（field）** 表示一个特征的向量
  -

$$V = F_{ReWeight}(A, E) = [a_1 \cdot e_1, \cdots, a_n \cdot e_n] = [v_1, \cdots, v_n] \tag{4}$$

Attention

SENet 通过对每个特征通道的自适应加权来建模通道之间的依赖关系，其权重代表了不同通道对于最终预测结果的重要性。Attention 则是通过计算特征之间的相关性来加权不同位置的特征，核心思想是为每个特征分配一个权重以反映该特征与其他特征之间的相对重要性。

Attention 包括：

- **Query（查询）**：用于表示当前需要关注的目标或查询的特征
- **Key（键）**：与Query进行匹配的特征，表示信息的内容或特征
- **Value（值）**：与Key相关联的实际信息

Attention公式可以描述为：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

计算步骤如下：

1. 计算Query和Key之间的相似度**Attention Score**，通常使用点积（Dot-Product）或加法（Additive）计算：

$$\text{score}(Q, K) = \frac{QK^T}{\sqrt{d_k}} \tag{6}$$

   其中，Q 是Query向量，K 是Key向量，d_k 是Key向量的维度，通常做缩放以防止点积过大。

2. 将相似度分数通过Softmax函数转换为概率分布（Attention权重）：

$$\text{Attention Weight} = \text{Softmax}(\text{score}(Q, K)) \tag{7}$$

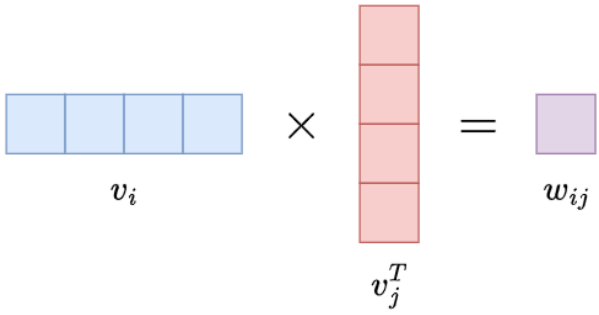
3. 通过加权平均**Value**，得到最终的输出：

$$\text{output} = \sum(\text{Attention Weight} \times V) \tag{8}$$

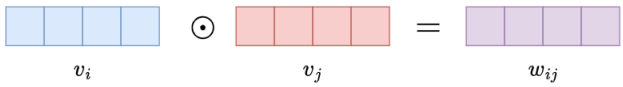
FiBiNet

FiBiNet 由SENet和交互层组成。其中，SENet实现了特征（field）加权，而交互层则实现了特征交叉。交互层是一个计算 **二阶特征交互** 的层（二阶交互描述了两个特征之间的相互作用，而非独立贡献），特征交互的方法有：

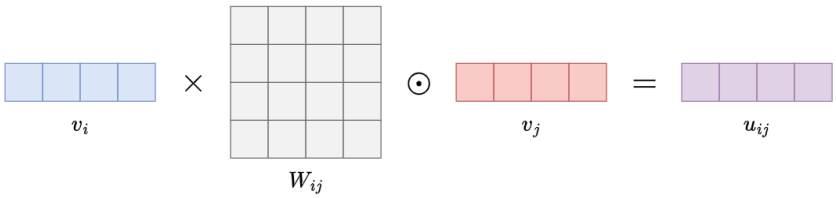
- **内积（Inner Product）**

- 两个向量相乘得到一个标量（实数）的运算，内积的几何意义是两个向量的乘积与它们夹角的关系
- 

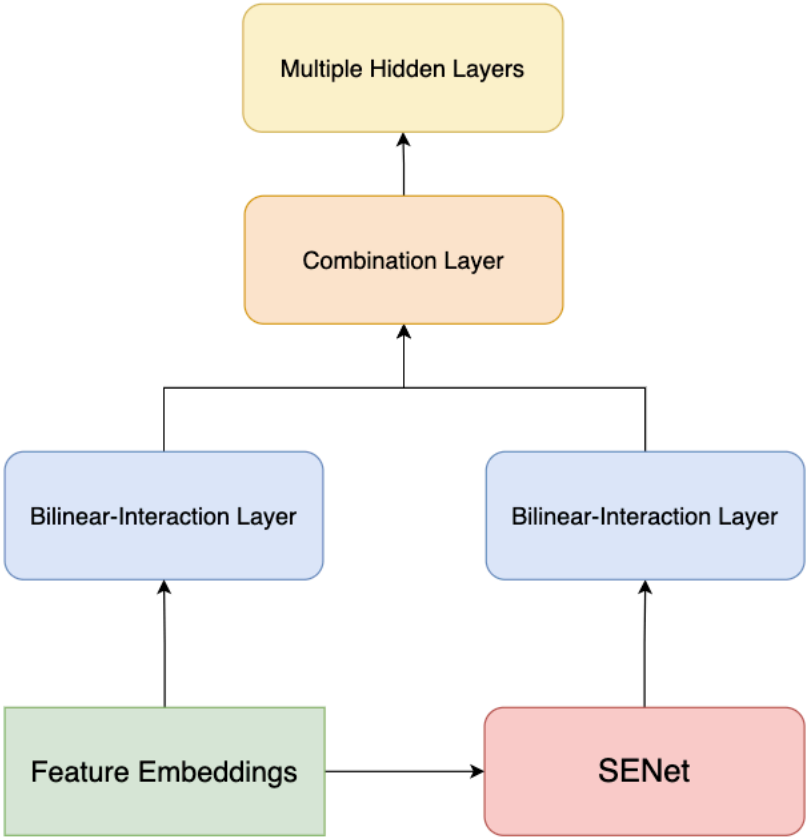
- $$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i \tag{9}$$

- **哈达玛积（Hadamard Product）**
  - 两个相同大小的矩阵（或向量）的每个对应元素进行逐一相乘，结果仍然是一个矩阵（或向量），其每个元素是原始矩阵（或向量）对应元素的乘积
  - 

  - $$\mathbf{a} \circ \mathbf{b} = [a_1 b_1, a_2 b_2, \cdots, a_n b_n] \tag{10}$$

- **双线性交互（Bilinear Interaction）**
  - 通过计算两个特征向量之间的双线性变换，捕捉它们的交互信息：$\text{Bilinear fnteraction} = \mathbf{x}^{\mathsf{T}} \mathbf{W} \mathbf{y}$
  - 其中 $\mathbf{W}$ 是权重矩阵，表示两个特征空间之间的交互关系
    - 若 $\mathbf{W}$ 是 **对角矩阵**，则 bilinear interaction 退化为元素级交互
    - 若 $\mathbf{W}$ 是 **稀疏矩阵**，可以高效地捕获重要的交互关系
    - 若 $\mathbf{W}$ 是 **全矩阵**，则可以完全表达两个向量的任意关系，但代价是更高的计算复杂度
  - 在FiBiNet中，双线性交互的实现方式是结合了内积和哈达玛积，如图：
  - 

FiBiNet 的模型结果如下：



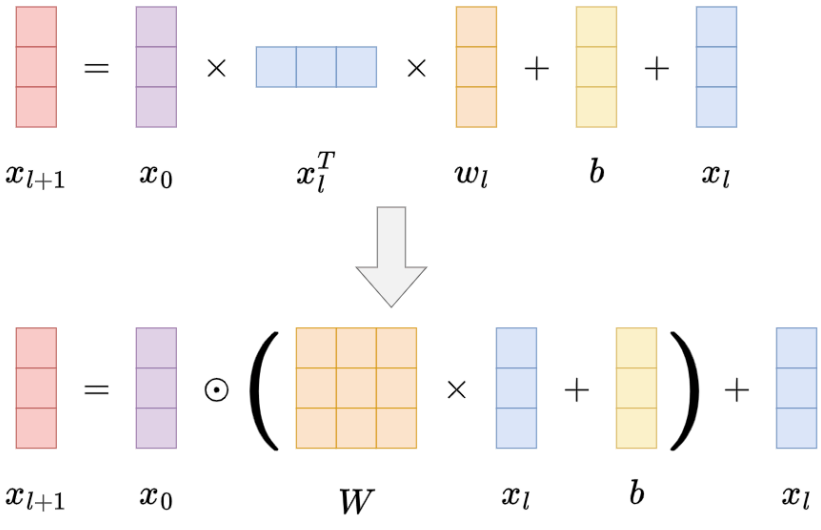
原始特征向量经过 Bilinear-Interaction 得到向量 p，另一路中原始向量经过 SENet 后在通过 Bilinear-Interaction 得到向量 q，将两个向量拼接（Concat）后得到向量 c 并给到下游 DNN 进行预测。

DCN

DCN（Deep & Cross Network）结合了 **深度学习（DNN）** 和 **显式的特征交叉（crossing）** 机制，能够有效地捕捉特征之间的高阶交互。DCN通过交叉层显式地学习特征交叉，而传统的DNN需要依靠全连接层隐式学习这些交互。DCN架构分为 Deep Network 和 Cross Network 两部分。

Cross Network 由 **交叉层（Cross Layer）** 构成，交叉层通过直接对输入特征进行交叉（即特征组合）来捕捉特征之间的高阶交互，不需要依赖全连接层逐层学习，从而显式建模输入特征的交互。

Cross Layer 经历了两版变化：



- 第一版：

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \mathbf{x}_l^T \mathbf{w}_l + \mathbf{b}_l + \mathbf{x}_l = f(\mathbf{x}_l, \mathbf{w}_l, \mathbf{b}_l) + \mathbf{x}_l \tag{11}$$

  - $x_0 \in \mathbb{R}^d$ 是包含原始特征的最底层输入
  - $x_l, x_{l+1} \in \mathbb{R}^d$ 分别表示第 l 层和第 $l+1$ 层的输出
  - $w_l, b_l \in \mathbb{R}^d$ 表示第 l 层的权重矩阵和偏置参数
  - 每个交叉层在特征交叉 f 后都会将其输入加回，即映射函数 $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ 拟合了 $x_{l+1} - x_l$ 的残差
- 第二版：

$$\mathbf{x}_{l+1} = \mathbf{x_0} \odot (W_l \mathbf{x}_l + \mathbf{b}_l) + \mathbf{x}_l \tag{12}$$

  - $x_0 \in \mathbb{R}^d$ 是包含原始特征的最底层输入
  - $x_l, x_{l+1} \in \mathbb{R}^d$ 分别表示第 l 层和第 $l+1$ 层的输出
  - $W_l \in \mathbb{R}^{d \times d}$ 和 $b_l \in \mathbb{R}^d$ 表示第 l 层的权重矩阵和偏置参数
  - $\odot$ 是哈达玛积运算（Hadamard Product），即逐元素相乘：两个相同大小的矩阵/向量的每个对应元素进行逐一相乘，结果仍是一个维度一致的矩阵/向量，其每个元素是原始矩阵/向量对应元素的乘积
- 第二版本的交叉层采用哈达玛积，向量 $w \in \mathbb{R}^d$ 替换为矩阵 $W \in \mathbb{R}^{d \times d}$，可以实现逐元素的加权，拟合能力加强
  - 第一版的交叉层学习的是特殊类型的高阶特征交互，其中每一层都是 x_0 的标量倍数（详细论证见 xDeepFM）：
    -

$$\begin{aligned} \mathbf{x}_{l+1} &= \mathbf{x}_0 \mathbf{x}_l^T \mathbf{w}_{l+1} + \mathbf{x}_l \\ &= \mathbf{x}_0 \left(\left(\alpha^l \mathbf{x}_0 \right)^T \mathbf{w}_{l+1} \right) + \alpha^l \mathbf{x}_0 \\ &= \alpha^{l+1} \mathbf{x}_0 \end{aligned} \tag{13}$$

    - 其中，

$$\alpha^{l+1} = \alpha^l (\mathbf{x}_0^T \mathbf{w}_{l+1} + 1) \tag{14}$$

    是一个标量，因此 x_{l+1} 仍然是 x_0 的一个标量倍数

DCN还保留了DNN的设计，即通过多层全连接层（FC层）来学习输入数据的非线性表示。这部分与标准的DNN相似，通过逐层的非线性变换来提取高层次的特征。

突破双塔

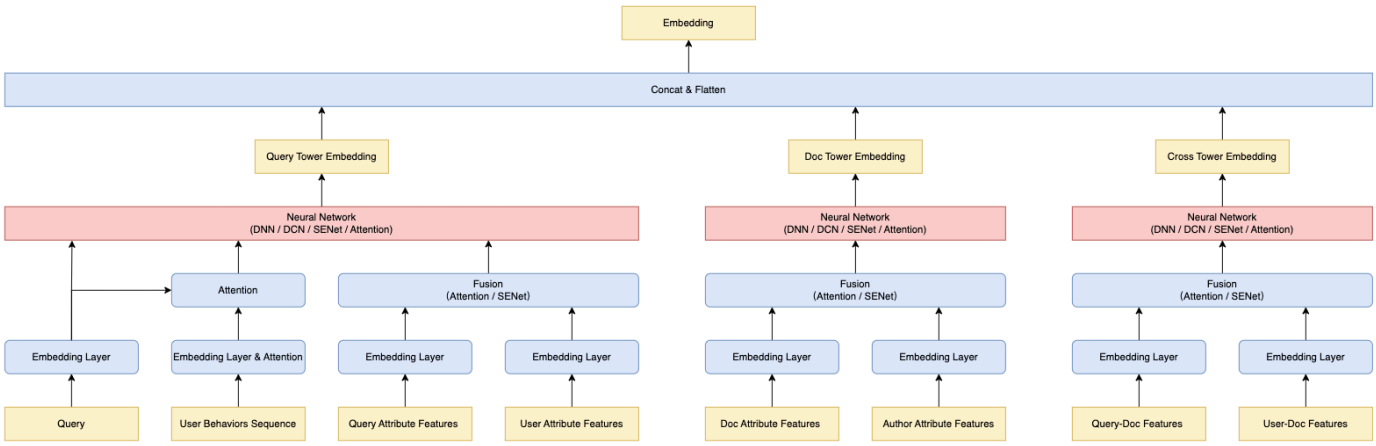
引入交叉塔

由于双塔模型缺乏 Query 和 Doc 的交互，为了增强模型对 Query 和 Doc 之间潜在复杂关系的理解，可以向粗排模型中引入交叉特征，并新增交叉塔（Cross Tower）。在线推理阶段，交叉塔和查询塔一同在线计算，交叉塔的输出向量与查询塔和文档塔的向量合并计算。

交叉塔特征输入

特征	类型
Query-Doc 文本匹配特征（token/NER/类目/作者名等）	Query-Doc 交叉特征
Query-Doc 互动特征（曝光/点击/点赞/收藏/评论等）	
User-Doc 匹配特征（User 画像-Doc 主题等）	User-Doc 交叉特征
User-Doc 互动特征（曝光/点击/点赞/收藏/评论等）	

三塔模型结构



引入后交互层

双塔模型采用内积计算相似度，采用 MLP 替换内积理论上可以建模更复杂的非线性关系。不过这种做法实际收益有限，因为双塔中每个塔的底层特征在向上传递的过程中，两个塔的细粒度的信息彼此未进行显式交叉，在逐层压缩的过程中已经损耗了信息。当通过增加输入特征或采用ResNet的方式给 MLP 层传递更多信息后，MLP 才能更大发挥其潜力，在一定程度上缓解 Query 和 Doc 特征的深度交互在隐藏层的早期阶段未被充分捕捉和表达的问题。

模型训练

样本选择偏差

粗排阶段通常需要在更大的候选集上进行排序，因此其训练数据应具有代表性，并覆盖更广泛的样本范围。而精排阶段则通常是对粗排后的候选集进行精细排序，这时的训练数据具有更强的相关性。如果训练样本完全采用搜索曝光点击数据，粗排模型可能只会关注那些在精排中已经被筛选出来的“优质”样本，从而忽视了很多潜在的优质样本。如此，粗排模型的召回能力会下降，而精排的收益却有限。

此外，如果粗排模型过度模仿精排的目标，它可能会过度优化那些已经在精排中排名靠前的文档，而忽视那些原本不太可能进入精排的文档。长期下去，将导致搜索系统对热门或常见文档的偏向，使得这些文档的曝光率不断提升，而冷门文档或新文档的曝光机会减少，带来严重的 **马太效应**。

粗排和精排的目标和策略应该保持一致性，同时避免在数据和目标设置上造成过度依赖，以促进系统的长期优化。

样本构建方法

训练样本的数据格式为 Query-Doc 对，通常由一个正样本和多个负样本构成。常见的样本挖掘方法主要有 **曝光样本**、**未曝光样本** 和 **随机负样本** 构成：

- **曝光点击-正样本**
  - 搜索日志中选取充足曝光下高交互率的样本作为 Query-Doc 正样本
- **精排过滤-负样本**
  - 进入精排阶段，因打分低被截断过滤的Doc（未曝光）与检索Query构建 Query-Doc 负样本
- **全库随机采样-负样本**
  - 在原始的全局物料库里，按 **曝光频率**、**类目相关** 分层随机抽取Doc和检索Query构建 Query-Doc 负样本
- **Batch内随机采样-负样本**
  - 在只包含正样本的一个训练Batch内，当前正样本 Query-Doc 中的Query与其他样本的Doc构建 Query-Doc 负样本

精排蒸馏粗排

由于计算性能和实时性要求，粗排模型通常需要采用较为简单的模型结构和较少的特征，为了弥补效果损失，通常采用蒸馏算法将精排模型知识迁移到粗排模型。

蒸馏算法

蒸馏算法通过最小化学生模型的输出与教师模型的输出之间的差距来训练学生模型。通常，采用两种损失函数：

- **硬标签损失（Hard Label Loss）**
  - 硬标签损失是基于真实标签（ground truth）计算的损失，通常使用交叉熵损失。硬标签损失 $\mathcal{L}_{\text{hard}}$ 为：

$$\mathcal{L}_{\text{hard}} = -\sum_{i} y_i \log(\hat{y}_{s,i}) \tag{15}$$

  - 其中：
    - y_i 是第 i 类的真实标签（通常是 one-hot 编码）
    - $\hat{y}_{s,i}$ 是学生模型预测的第 i 类的概率
- **软标签损失（Soft Label Loss）**
  - 软标签损失通过教师模型的输出（即软标签）来计算，这些软标签是教师模型在推理过程中产生的概率分布（softmax 输出）。软标签损失 $\mathcal{L}_{\text{soft}}$ 通常使用 **KL 散度（Kullback-Leibler Divergence）** 来度量学生模型的输出与教师模型的输出分布之间的差异。具体的公式为：

$$\mathcal{L}_{\text{soft}} = D_{\text{KL}}(\hat{y}_t \parallel \hat{y}_s) = \sum_{i} \hat{y}_{t,i} \log\left(\frac{\hat{y}_{t,i}}{\hat{y}_{s,i}}\right) \tag{16}$$

  - 其中：
    - $\hat{y}_{t,i}$ 是教师模型对第 i 类的预测概率
    - $\hat{y}_{s,i}$ 是学生模型对第 i 类的预测概率
- **总损失函数**
  - 最终损失函数是硬标签损失和软标签损失的加权和：

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{hard}} + \beta \cdot \mathcal{L}_{\text{soft}} \tag{17}$$

  - 其中：
    - α 和 β 是超参数，用于控制硬标签损失和软标签损失的权重
    - $\mathcal{L}_{\text{hard}}$ 是基于真实标签的交叉熵损失
    - $\mathcal{L}_{\text{soft}}$ 是基于教师模型输出的软标签损失（KL 散度）

- **温度调节（Temperature Scaling）**
  - 为了让学生模型更好地学习教师模型的输出分布，通常对教师模型的 softmax 输出进行温度调节。引入温度参数 T 后，softmax 函数变为：

$$\hat{y}_{t,i} = \frac{e^{z_{t,i}/T}}{\sum_j e^{z_{t,j}/T}} \tag{18}$$

  - 其中：
    - $z_{t,i}$ 是教师模型的原始 logits 输出（即模型的未归一化预测）
    - T 是温度参数，通常 $T > 1$，增加温度后，softmax 输出的概率分布变得更加平滑
    - $\hat{y}_{t,i}$ 是教师模型经过温度调节后的输出概率

优化目标

多目标预估

引入多个搜索互动行为作为优化目标（如点击、点赞、收藏、分享、关注、评论、截图、成交等）能够使得模型在训练过程中同时考虑多个维度的反馈，从而在搜索结果排序时，能够更加全面地反映用户的真实需求和行为模式。

设模型输出为 $\hat{y}_i$，表示对某个候选项 i 的预测评分。目标函数 L 可以写成多任务学习的形式：

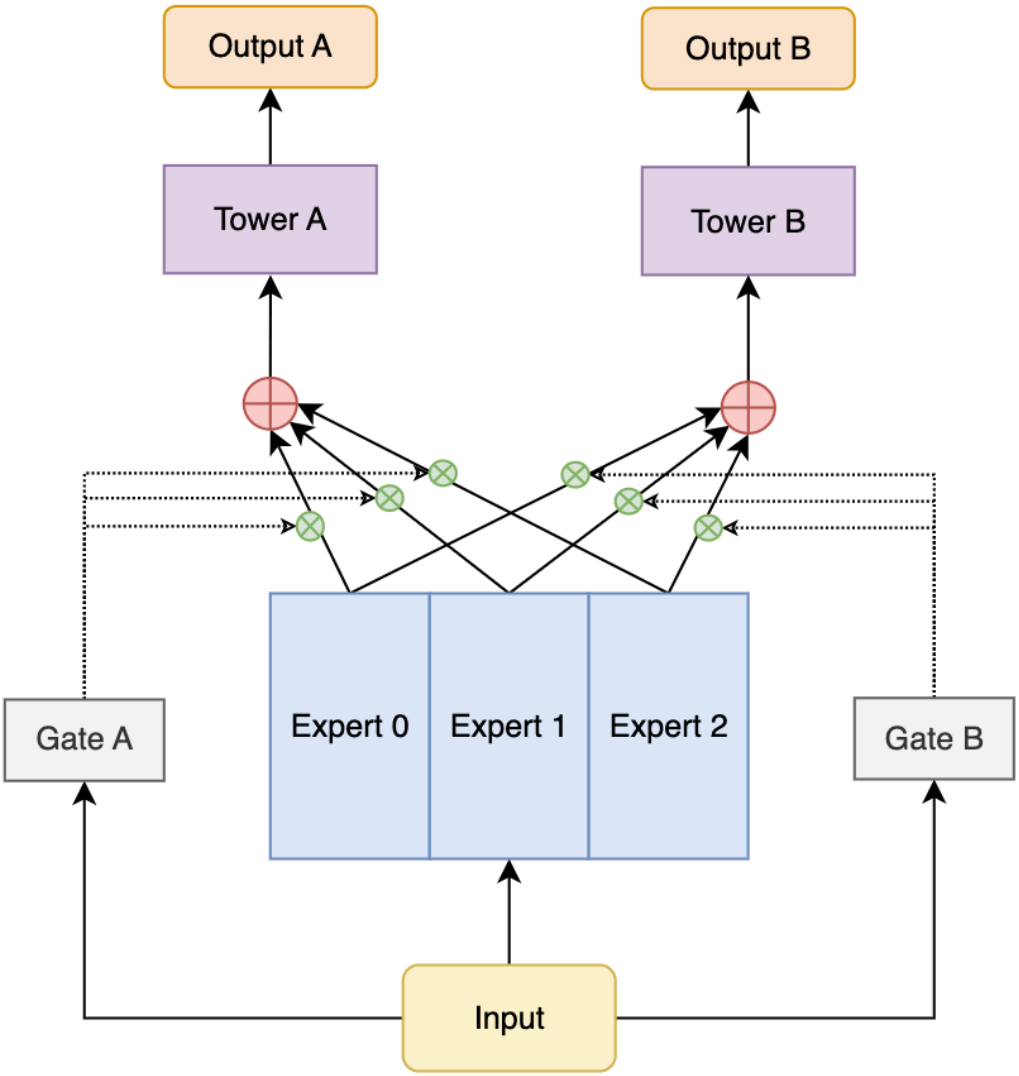
$$L = \lambda_c L_c(\hat{y}_i, Y_c) + \lambda_l L_l(\hat{y}_i, Y_l) + \lambda_s L_s(\hat{y}_i, Y_s) + \lambda_r L_r(\hat{y}_i, Y_r) \tag{19}$$

其中：

- L_c，L_l, L_s, L_r 分别表示点击、点赞、收藏、评论的损失函数（BCE Loss）
- λ_c，λ_l，λ_s，λ_r 是各个目标的权重参数，用于平衡不同目标的贡献

MMoE

常见的多目标预估模型有MMoE（Multi-gate Mixture of Experts），任务 T_k 的最终输出由多个共享专家网络 $(e_1, e_2, \ldots, e_N)$ 的加权组合生成，权重由任务特定的门控网络 g_k 动态计算。



专家网络

每个专家网络的输入为通用特征 x，经过专家网络 e_i 后得到输出 h_i：

$$h_i = e_i(x), \quad i = 1, 2, \ldots, N \tag{20}$$

其中：

- e_i 是第 i 个专家网络
- $h_i \in \mathbb{R}^d$ 是第 i 个专家网络的输出

门控网络

任务 T_k 的门控网络根据输入特征 x，计算每个专家的权重 $g_{k,i}$：

$$g_{k,i} = \text{softmax}(W_k \cdot x + b_k)_i \tag{21}$$

其中：

- $g_{k,i} \in [0, 1]$ 是专家 e_i 对任务 T_k 的权重
- W_k 和 b_k 是门控网络的参数

专家输出融合

任务 T_k 的最终共享特征输出 o_k 是所有专家输出的加权和：

$$o_k = \sum_{i=1}^{N} g_{k,i} \cdot h_i \tag{22}$$

其中：

- $g_{k,i}$ 是任务 T_k 对专家 e_i 的权重
- h_i 是专家 e_i 的输出

任务预测

共享特征 o_k 传递到任务特定的预测网络 f_k，得到最终的任务预测结果：

$$\hat{y}_k = f_k(o_k) \tag{23}$$

其中 f_k 是任务 T_k 的特定网络。

损失函数

Pointwise Loss

Pointwise Loss 只关注单个候选项（如分类或回归任务）。

交叉熵（Cross Entropy，CE）

在粗排实现目标预估时，每个目标被定义为 **二分类** 任务，输出结果为各互动行为的概率。分类任务采用交叉熵损失，交叉熵主要用于衡量两个概率分布之间的差异：

$$\mathrm{CE}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i) \tag{24}$$

- y_i 是真实标签的独热编码向量，第 i 类的标签为 1，其它类别为 0
- $\hat{y}_i$ 是模型对于第 i 类的预测概率

Pairwise Loss

Pairwise Loss 关注候选项之间的两两比较。

对比损失

对比损失衡量两个文档的相对排序是否正确。如果文档对的相关性排序是正确的，那么损失应该较小；如果排序不正确，则损失应该较大：

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{2}\left[y \cdot d^2 + (1-y) \cdot \max(0, m-d)^2\right] \tag{25}$$

- y 是文档对的标签，取值为 0 或 1，表示文档 D_1 是否应该排在文档 D_2 之前
- d 是文档 D_1 和 D_2 在模型预测中的距离（通常是两者的评分差）
- m 是一个超参数，表示 “margin”（边际），即在没有损失的情况下，文档的评分差需要达到的最小值

RankNet损失

RankNet 使用 **对数损失** 来训练排序模型。RankNet 通过概率的形式预测文档的相对排序，给定一对文档 D_1 和 D_2，模型输出一个概率，表示 D_1 比 D_2 相关的概率：

$$\mathcal{L}_{\text{RankNet}} = -\sum_{i,j} \left[p_{ij} \log \hat{p}_{ij} + (1 - p_{ij}) \log(1 - \hat{p}_{ij}) \right] \tag{26}$$

- p_{ij} 是文档对 D_1 和 D_2 的真实排序标签（如果 D_1 应该排在 D_2 前面，则 $p_{ij} = 1$，否则为 0）
- $\hat{p}_{ij}$ 是模型预测 D_1 排在 D_2 前面的概率

Listwise Loss

Listwise Loss 直接优化一个完整候选列表的排名质量。

LambdaRank

LambdaRank 在 RankNet 的基础上进一步发展，旨在直接优化排序指标（如 NDCG）而设计。

目标函数

LambdaRank 的核心在于设计与排名指标相关的**动态梯度权重**，以高效优化目标。

1. **NDCG 指标**

$$\text{NDCG@k} = \frac{\sum_{i=1}^{k} \frac{2^{y_i} - 1}{\log_2(i+1)}}{\text{IDCG@k}} \tag{27}$$

   - y_i：文档 i 的相关性标签（通常是人工标注或规则确定的相关性得分）
   - i：文档在当前排序中的位置
   - IDCG@k：理想情况下的 DCG 值（Ideal DCG）

2. **梯度定义**
   对于候选文档对 (i, j)，假设 $y_i > y_j$（即文档 i 比 j 更相关），对应的梯度更新为：

$$\lambda_{ij} = |\Delta Z_{ij}| \cdot \sigma(s_j - s_i) \tag{28}$$

   - ΔZ_{ij}：交换文档 i 和 j 后，排名指标（如 NDCG）的增量变化

$$\Delta Z_{ij} = \left| \frac{1}{\log_2(1 + \text{rank}_i)} - \frac{1}{\log_2(1 + \text{rank}_j)} \right| \cdot |2^{y_i} - 2^{y_j}| \tag{29}$$

   - $\sigma(s_j - s_i)$：模型预测的分数差对应的 Sigmoid 函数

$$\sigma(s_j - s_i) = \frac{1}{1 + \exp(s_i - s_j)} \tag{30}$$

损失函数

LambdaRank 的损失函数是基于梯度权重的更新，而不是显式地定义为一个封闭公式。但可以通过梯度描述模型的优化方向：

$$\mathcal{L}_{\text{LambdaRank}} = \sum_{i,j} \lambda_{ij} \cdot \log(\sigma(s_i - s_j)) \tag{31}$$

梯度更新

对文档 i 的得分 s_i 的梯度更新规则为：

$$\frac{\partial \mathcal{L}}{\partial s_i} = \sum_{j \neq i} (\lambda_{ij} - \lambda_{ji}) \tag{32}$$

- 如果 $y_i > y_j$，则 $\lambda_{ij} > 0$，模型需要提升 s_i
- 如果 $y_i < y_j$，则 $\lambda_{ij} < 0$，模型需要降低 s_i

总结

粗排在搜索链路中的定位介于召回与精排之间，既要承接召回的多样性，又要为精排提供足够的排序精度支撑。在设计粗排策略时，需要明确其与精排的差异化目标，从子集优质性、协作效率、覆盖性等方面入手，通过特征优化、多目标建模等手段，在效率与效果之间找到平衡点，从而最大化搜索系统的整体表现。

参考文献

1. Embedding-based Retrieval in Facebook Search
2. Embedding-based Product Retrieval in Taobao Search
3. Multi-Objective Personalized Product Retrieval in Taobao Search
4. MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search
5. FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction
6. Dense Text Retrieval based on Pretrained Language Models: A Survey
7. Deep Interest Network for Click-Through Rate Prediction
8. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems
9. Deep & Cross Network for Ad Click Predictions
10. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems
11. Squeeze-and-Excitation Networks
12. Notes on Noise Contrastive Estimation and Negative Sampling
13. Learning Tree-based Deep Model for Recommender Systems
14. Approximate Nearest Neighbor Search under Neural Similarity Metric for Large-Scale Recommendation
15. Beyond Two-Tower Matching: Learning Sparse Retrievable Cross-Interactions for Recommendation
16. Deep Retrieval: An End-to-End Learnable Structure Model for Large-Scale Recommendations
17. I^3 Retriever: Incorporating Implicit Interaction in Pre-trained Language Models for Passage Retrieva

18. A Dual Augmented Two-tower Model for Online Large-scale Recommendation
19. Multivariate Representation Learning for Information Retrieval
20. Beyond Two-Tower: Attribute Guided Representation Learning for Candidate Retrieval
21. Multi-Aspect Dense Retrieval
22. VIRT: Improving Representation-based Text Matching via Virtual Interaction
23. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts
24. Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations