[搜索系统算法解析-OAA]

重排

重排 是精排后的一个阶段，主要负责在最终展示结果前对精排后的排序列表进行进一步优化和调整（微调）。重排核心目标是保证一定相关性的前提下，提高结果的多样性，从而提升用户体验，满足用户在不同方面的需求，避免搜索结果过于单一、相似和同质化，为用户提供更丰富、全面的信息。

单点估计方法

分桶

1. 将搜索结果按照文档属性（如类目、来源、形态、样式等）进行分桶
2. 每个桶中的文档按照精排打分降序排序后，再按照指定规则从不同桶中交替抽取结果展示

滑动窗口

1. 将候选文档按精排打分从高到低降序排序
2. 确定滑动窗口的大小 N（窗口大小可以是固定的，或根据实际情况动态调整）
3. 选择前 N 个候选项，作为滑动窗口中的初始内容
4. 根据业务需求，设置打散规则，确保滑动窗口内的内容多样化，避免相似内容集中在一起。具体的，对滑动窗口中的内容，检查其是否满足多样性约束。如果不满足，则通过交换、重排序或替换等方式调整窗口内的文档，直到满足多样性要求为止
5. 窗口向右滑动一个单位，去掉窗口左边的第一个结果，并加入窗口右侧新的结果
6. 逐步滑动窗口并优化多样性，直到所有搜索结果都被处理完毕，或者达到预设的多样性目标

MMR

设有一组候选文档 $D = \{d_1, d_2, \ldots, d_m\}$，需要从中选择一组结果 $S \subseteq D$，其中每个结果 d_i 都有一个与查询 Q 的相关性评分（精排模型打分） $\mathrm{Rel}(d_i)$，在满足相关性的同时要避免选择与已选结果 S 中内容过于相似的文档。**MMR（Maximal Marginal Relevance）** 打分函数为：

$$\mathrm{MMR}(d) = \lambda \cdot \mathrm{Rel}(d) - (1 - \lambda) \cdot \max_{d' \in S} \mathrm{Sim}(d, d') \tag{1}$$

其中：

- $\mathrm{Rel}(d)$ 是文档 d 与查询 Q 的相关性得分
- $\mathrm{Sim}(d, d')$ 是文档 d 与已选结果集 S 中某个文档 d' 的相似度（可以用余弦相似度或其它度量方法计算）
- λ 是一个权重参数，用于平衡相关性和多样性之间的权衡，通常 $0 \leq \lambda \leq 1$
  - 如果 $\lambda = 1$，MMR算法完全关注相关性
  - 如果 $\lambda = 0$，MMR算法完全关注多样性（最小化相似度）
- S 是已选结果集，即在当前选择阶段之前已经选择的结果集合

DPP

DPP（Determinantal Point Processes），行列式点过程 是一种用于选择和排序多样化子集的概率模型，其通过最大化子集的行列式来鼓励选择之间的多样性。DPP 本质是一种在给定集合中选择子集的概率分布，能够有效地避免相似项的集中出现，从而实现多样性的优化。

相似度矩阵

给定一个候选集合 $X = \{x_1, x_2, \ldots, x_n\}$，每个元素 x_i 对应一个文档对象。DPP基于 **相似度矩阵** 来建模对象之间的关系。矩阵 K 是一个对称矩阵，其中每个元素 K_{ij} 表示对象 x_i 和 x_j 之间的相似度。

- **相似度矩阵**：

$$K = \begin{pmatrix} K_{11} & K_{12} & \ldots & K_{1n} \\ K_{21} & K_{22} & \ldots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \ldots & K_{nn} \end{pmatrix} \tag{2}$$

  其中，K_{ij} 是项 x_i 和项 x_j 之间的相似度。

更具体的，将 DPP 模型应用于搜索/推荐任务，需要构建**核矩阵**。核矩阵可以写成 Gram 矩阵 $K = B^\top B$，其中 B 的列是表示文档/物品的向量。将每个列向量 B_i 构建为文档相关性得分 $r_i \geq 0$ 与一个归一化向量 $f_i \in R^D$（满足 $\|f_i\|_2 = 1$）的乘积。核矩阵 K 中的元素可以写成：

$$K_{ij} = \langle B_i, B_j \rangle = \langle r_i f_i, r_j f_j \rangle = r_i r_j \langle f_i, f_j \rangle \tag{a}$$

其中 $\langle f_i, f_j \rangle = S_{ij}$ 为衡量文档 i 和文档 j 之间相似性的数值。因此，用户查询 u 的核矩阵可以写成：

$$K = \mathrm{Diag}\,(r_u) \cdot S \cdot \mathrm{Diag}\,(r_u) \tag{3}$$

其中 $\mathrm{Diag}\,(r_u)$ 是对角线向量为 r_u 的对角矩阵。

K_{R_u} 表示被文档子集 R_u 索引的核矩阵的子矩阵，衡量文档子集 R_u 的相关性和多样性的对数概率为：

$$\log\det\,(K_{R_u}) = \sum_{i \in R_u} \log\left(r_{u,i}^2\right) + \log\det\,(S_{R_u}) \tag{b}$$

等式 (b) 中等号的右边两项分别衡量相关性和多样性，第二项在 R_u 的文档表示正交时最大化（促进多样性）。

> **核矩阵**（或称为**Gram矩阵**）是一个对称矩阵，其中的元素表示数据点对之间的相似性。给定数据集 $X = \{x_1, x_2, \ldots, x_n\}$，核矩阵 K 的元素 K_{ij} 表示数据点 x_i 和 x_j 之间的相似度，通常通过核函数（Kernel Function）计算：
>
> $$K_{ij} = k(x_i, x_j) \tag{4}$$
>
> 其中，$k(x_i, x_j)$ 是核函数，表示数据点 x_i 和 x_j 之间的相似性度量。
>
> - 线性核：$k(x_i, x_j) = x_i^T x_j$
> - 高斯核（RBF核，Radial Basis Function）：$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$
> - 多项式核（Polynomial Kernel）：$k(x_i, x_j) = (x_i^T x_j + c)^d$
> - Sigmoid核：$k(x_i, x_j) = \tanh(\alpha x_i^T x_j + c)$

DPP概率模型

DPP 能够将复杂的概率计算转换成简单的行列式计算，基于 DPP 的多样性算法通过计算核矩阵的行列式找到候选中相关性和多样性最大的子集。

如上给定一个候选集合 $X = \{x_1, x_2, \ldots, x_n\}$，并给定一个子集 $S \subseteq X$。行列式点过程 P 是在 X 的所有子集的集合 2^X 上的概率测度。当 P 对空集赋予非零概率时，存在一个矩阵 $K \in R^{n \times n}$ 使得每个子集 $S \subseteq X$，S 的概率为：

$$\mathbb{P}(S) \propto det(K_S) \tag{5}$$

其中 K 为一个实数半正定矩阵核矩阵，由 X 的元素索引。

将上述公式归一化后，DPP 选取的概率由以下公式给出：

$$\mathbb{P}(S) = \frac{\det(K_S)}{\sum_{S' \in \mathbb{X}} \det(K_{S'})} = \frac{\det(K_S)}{\det(K + I)} \tag{6}$$

- K_S：是子集 S 中所有项之间的相似度矩阵。即，如果 $S = \{x_1, x_3, x_4\}$，那么 K_S 是矩阵 K 的一个子矩阵，只包含 x_1, x_3, x_4 之间的相似度
- $\mathbb{X}$：是 X 中所有子集的集合
- $\det(K_S)$：是子集 S 的行列式，行列式越大，表示子集内项之间的多样性越高。即子集内的项之间越不相似，行列式的值就越大
- $\det(K + I)$：是对矩阵 K 加上单位矩阵 I 之后的行列式

> **半正定矩阵（Positive Semi-Definite Matrix）：** 给定实对称矩阵 $A \in \mathbb{R}^{n \times n}$，若对于任何非零向量 $\mathbf{x} \in \mathbb{R}^n$，都有：
>
> $$\mathbf{x}^T A \mathbf{x} \geq 0 \tag{7}$$
>
> 则矩阵 A 为半正定矩阵。 半正定矩阵的特征值是非负的，如果矩阵 A 的所有特征值都是正的，那么它被称为 **正定矩阵**（Positive Definite Matrix）。
> 半正定矩阵通常与向量空间中的二次型相关联。具体地，对于矩阵 A 和向量 $\mathbf{x}$，二次型 $\mathbf{x}^T A \mathbf{x}$ 代表一个标量，描述了向量 $\mathbf{x}$ 在矩阵 A 所定义的度量空间下的 “大小”。 如果 A 是半正定的，意味着任何向量在该空间中都不会导致负值，因此矩阵 A 没有 “反向” 的作用，也就是说，它没有 “使向量反向拉伸” 的作用。半正定矩阵对应的二次型是一个凸函数，几何上可以理解为在定义该函数的空间中，每个切面都是向上的，这样的函数值总是保持非负值。

最大化行列式

在 DPP 模型中，每次选择一个子集时，选择的概率取决于该子集内元素的相似度。为了最大化多样性，DPP 会通过最大化子集的行列式来选择最优子集。即在候选文档集合中选取可以最大化后验概率的文档子集，DPP 的 **最大后验概率（MAP）** 推断可被描述为：

$$S_{\text{map}} = \arg\max_{S \subseteq X} \det(K_S) \tag{8}$$

如上，选择子集的过程为在给定的候选集合 X 中，通过选择一个子集 S 来 **最大化其行列式**，即最大化 $\det(K_S)$，确保子集的多样性尽可能高。

> - **行列式的几何解释：** 行列式表示由矩阵的列（或行）所张成的向量空间的体积。如果将矩阵的列向量（或行向量）看作是空间中的向量，那么行列式的大小与这些向量的排列密切相关。

    - **行列式为0**：如果矩阵的行列式为零，表示矩阵的列向量（或行向量）是线性相关的，即这些向量不张成一个高维空间
    - **行列式大于0**：如果矩阵的行列式大于零，表示矩阵的列向量在特征空间中是**线性独立**的，即它们占据了不同的维度，能够覆盖一个较大的空间
- **线性独立性与多样性**：如果多个向量是线性独立的，那么它们无法通过线性组合来表示其他向量，意味着这些向量在空间中占据了不同的维度。
    - **高行列式 → 线性独立**：当矩阵的行列式较大时，意味着矩阵中的列向量（或行向量）是线性独立的，说明这些向量在特征空间中分布得较广泛
    - **低行列式 → 线性相关**：当行列式较小或为零时，矩阵中的列向量是线性相关的，这意味着这些向量之间的相似度很高，不能有效地覆盖多维空间，导致内容的多样性较低

MAP推断

DPP 的最大后验概率（MAP）推断是一个 NP-Hard 问题，DPP 通常使用 **贪心算法** 来逐步选择元素，直到达到所需的子集大小。即每次迭代中，选择文档 j 添加到结果集合 S_g 中：

$$j = \arg\max_{i \in Z \setminus S_g} \log\det(K_{S_g \cup \{i\}}) - \log\det(K_{S_g}) \tag{1}$$

由于 K 是半正定矩阵，其所有主子矩阵也是半正定矩阵。假设 $\det(K_{S_g}) > 0$，K_{S_g} 的Cholesky分解为 $K_{S_g} = VV^T$，其中 V 是可逆下三角矩阵。对于任何 $i \in Z \setminus S_g$，$K_{S_g \cup \{i\}}$ 的Cholesky分解可以推导为：

$$K_{S_g \cup \{i\}} = \begin{bmatrix} K_{S_g} & K_{S_g,i} \\ K_{i,S_g} & K_{ii} \end{bmatrix} = \begin{bmatrix} V & 0 \\ c_i & d_i \end{bmatrix} \begin{bmatrix} V & 0 \\ c_i & d_i \end{bmatrix}^\top \tag{2}$$

其中行向量 c_i 和标量 $d_i \geq 0$ 满足：

$$Vc_i^\top = K_{S_g,i}, \tag{3}$$

$$d_i^2 = K_{ii} - \|c_i\|_2^2 \tag{4}$$

根据等式 (2) 可以推导出

$$\det(K_{S_g \cup \{i\}}) = \det(VV^\top) \cdot d_i^2 = \det(K_{S_g}) \cdot d_i^2 \tag{5}$$

因此，(1) 等式可以简化为：

$$j = \arg\max_{i \in Z \setminus Y_g} \log(d_i^2) \tag{6}$$

等式 (6) 被求解后，根据等式 (2)，$K_{S_g \cup \{j\}}$ 的Cholesky分解变为

$$L_{Y_g \cup \{j\}} = \begin{bmatrix} V & 0 \\ c_j & d_j \end{bmatrix} \begin{bmatrix} V & 0 \\ c_j & d_j \end{bmatrix}^\top \tag{7}$$

其中 c_j 和 d_j 是现成可用的。因此，可以在向 S_g 添加新项目（文档）后有效更新 K_{S_g} 的Cholesky因子。

对于每个项目 i，c_i 和 d_i 也可以增量更新。定义 c_i' 和 d_i' 为 $i \in Z \setminus (S_g \cup \{j\})$ 的新向量和标量。则有：

$$\begin{bmatrix} V & 0 \\ c_j & d_j \end{bmatrix} \begin{bmatrix} c_i' \\ d_i' \end{bmatrix}^\top = K_{S_g \cup \{j\},i} = \begin{bmatrix} K_{S_g,i} & K_{ji} \end{bmatrix} \tag{8}$$

结合等式 (8) 和 (3)：

$$c_i' = [c_i \quad (K_{ji} - \langle c_j, c_i \rangle)/d_j] \doteq [c_i \quad e_i] \tag{9}$$

则等式（4）为：

$$d_i'^2 = K_{ii} - \|c_i'\|_2^2 = K_{ii} - \|c_i\|_2^2 - e_i^2 = d_i^2 - e_i^2 \tag{9}$$

最初，$Y_g = \emptyset$，并且根据等式 (5)，$d_i^2 = \det(K_{ii}) = K_{ii}$。对于无约束MAP推断，停止条件是 $d_{2j} < 1$，或者当施加基数约束时 $S_g > N$。对于后者，引入小数 $\varepsilon > 0$ 并添加 $d_2^j < \varepsilon$ 到停止标准中，以计算 $1/d_j$ 的数值稳定性。

PRM

PRM（Generative Rerank Network）采用 Transformer 结构实现个性化重排。具体的，PRM 模型由输入层、编码层和输出层组成，输入层将初始列表中的所有文档/物品的特征向量进行编码，编码层使用 Transformer 结构来捕捉文档/物品间的相互影响，输出层生成每个文档/物品的点击概率。

PRM 的输入为原始特征矩阵 $X \in R^{n \times d_{\text{feature}}}$ 和个性化矩阵 $PV \in R^{n \times d_{\text{pv}}}$ 拼接（PV 取精排模型的倒数第二层的输出向量），同时利用初始列表中的顺序信息，将排名的向量表示 $PE \in R^{n \times (d_{\text{feature}} + d_{pv})}$ 注入输入向量中：

$$E = \begin{bmatrix} x_{i_1}; pv_{i_1} \\ x_{i_2}; pv_{i_2} \\ \cdots \\ x_{i_n}; pv_{i_n} \end{bmatrix} + \begin{bmatrix} pe_{i_1} \\ pe_{i_2} \\ \cdots \\ pe_{i_n} \end{bmatrix} \tag{10}$$

使用一个简单的前馈网络将特征矩阵 $E \in R^{n \times (d_{\text{feature}} + d_{pv})}$ 转换为 $E \in R^{n \times d}$，即 $E = EW^E + b^E$。

PEM 的输出层使用一个线性层后接一个 softmax 层来为每个项目 $i = i_1, \ldots, i_n$ 生成一个分数 $\text{Score}(i)$ 来对项目进行一步重新排序：

$$\text{Score}(i) = \text{softmax}\left(F^{(N_x)} W^F + b^F\right), i \in \mathcal{S}_r \tag{11}$$

其中 $F^{(N_x)}$ 是 N_x 个 Transformer 编码器块的输出。W^F 是可学习的投影矩阵，b^F 是偏置项。

上下文感知建模

MMR、DPP 和 PRM 都属于单点估计，即假设候选内容的评分独立，通过每一步的贪心选择实现候选列表的生成，这种方式未考虑已选中内容对当前候选内容的影响。

基于上下文感知的排序模型采用 **序列生成-序列评估** 的架构，通过建模序列生成过程动态调整排序决策，以全局最优的视角实现重排。

miRNN

miRNN（mutual influence RNN）将排序问题视为序列生成问题，使用循环神经网络（RNN）模型来估计点击/购买概率。RNN模型通过迭代计算每个文档/商品的点击/购买概率，并使用 beam search 算法序列生成最优排序。

设 $S = (1, \cdots, N)$ 为待排序的文档集合，O 表示 S 的所有排列集合，$o = (o_1, \ldots, o_N) \in O$ 是一个排列，用户在 o 中点击文档 i 的概率表示如下：

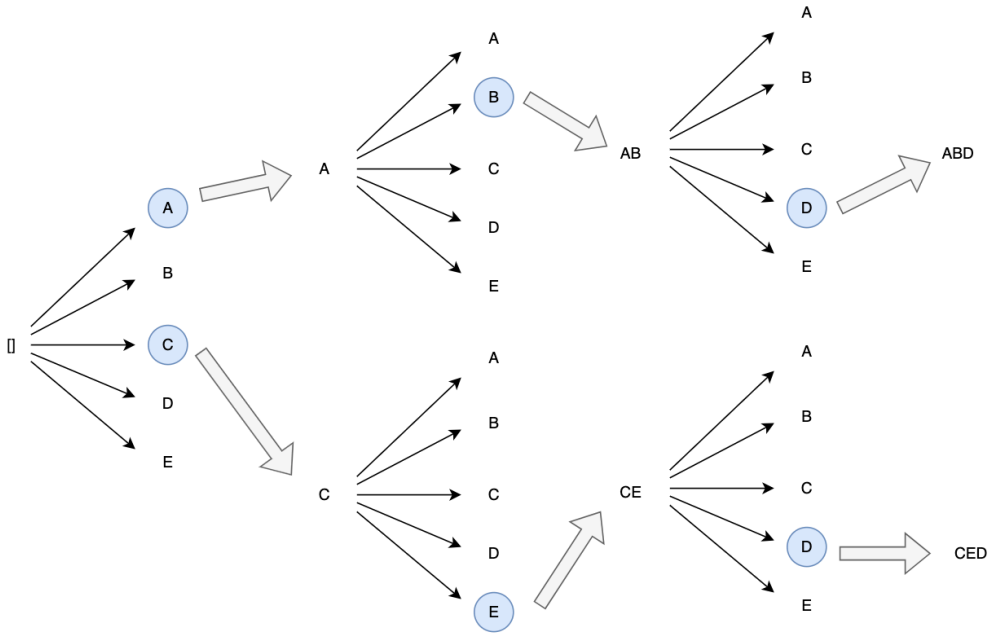
$$
\begin{aligned}
p(o_i|o_1,\ldots,o_{i-1}) &= \text{sigmoid}(W_h h_i) \\
h_i &= \text{RNN}(h_{i-1}, f(o_i))
\end{aligned} \tag{12}
$$

其中，h_{i-1} 是序列 $[o_1,\ldots,i-1]$ 的 RNN 向量（上下文表示），$f(o_i)$ 表示提取当前排列为 o_i 的文档特征。

Beam Search

MMR 和 DPP 都是基于贪心策略（Greedy Search），在每一步仅选择当前状态下使目标效用函数（如相关性与多样性的平衡）最大化的文档加入候选列表。贪心策略适合快速生成候选结果，但当对全局最优解有更高要求时，无法达到整体最优解。

Beam Search 是一种启发式搜索算法，用于在序列生成过程中保留多个最优候选，以避免局部最优解。在每个步骤中，算法保留 k 个具有最高评分的部分序列（称为 beam width 或 beam size），并基于这些部分序列扩展生成新的候选，直至达到所需的序列长度。

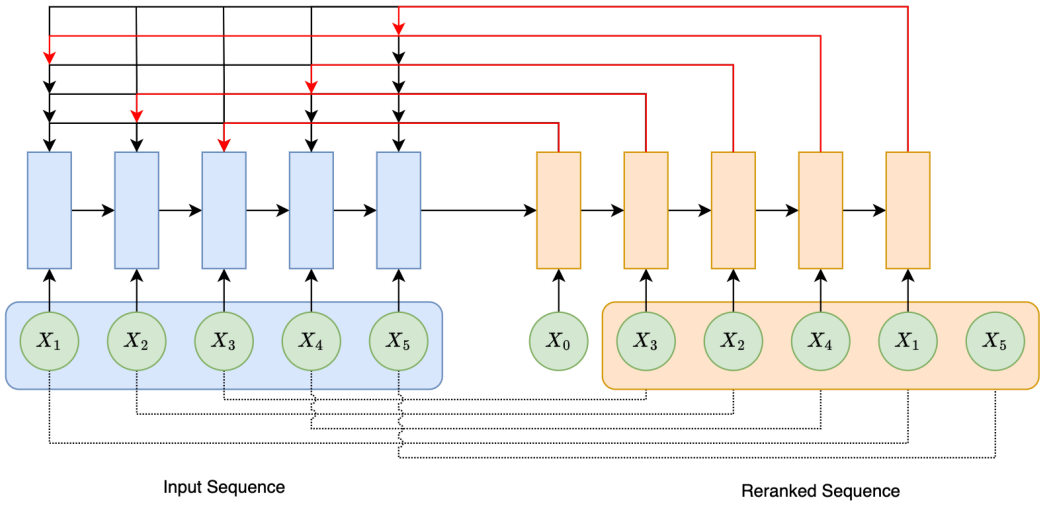


1. **初始化**
   - 设定 beam size 为 k
   - 初始化空序列集合 $B_0 = \{[]\}$
2. **迭代生成序列**
   - 对于每个时间步 $t = 1, 2, \ldots, L$：
     - **扩展候选序列**: 对于当前的每个部分序列 $s_{1:t-1} \in B_{t-1}$ 和每个候选内容 $x \in X \setminus s_{1:t-1}$，生成新的序列 $s_{1:t} = [s_{1:t-1}, x]$
     - **计算序列得分**: 为每个新序列 $s_{1:t}$ 计算得分函数 $\text{Score}(s_{1:t})$，该得分函数考虑序列的相关性和多样性
     - **选择 top-k 序列**: 从所有新生成的序列中选择得分最高的 k 个，构成新的部分序列集合 B_t
3. **终止条件**
   - 当达到预定的序列长度 L 或满足其他终止条件时，停止迭代
   - 从最终的序列集合 B_L 中选择得分最高的序列作为输出

Seq2slate

Seq2Slate 的核心是 Seq2Seq 架构，利用循环神经网络（RNN）实现，模型输入是候选集合列表，输出是排序后的结果。Seq2Slate 包含编码器和解码器两个部分：

- **编码器（Encoder）：** 将输入的项目序列编码为上下文向量
- **解码器（Decoder）：** 解码器则根据该上下文向量逐步生成新的项目序列。在每一步，解码器通过指针网络（Pointer Network）选择下一个最优项目，形成新的排序



Seq2Slate 流程可以描述为：

1. **编码器**
   输入候选集合 $X = \{x_1, x_2, \ldots, x_n\}$：

$$h_i = \text{Encoder}(x_i), \quad i \in \{1, 2, \ldots, n\} \tag{13}$$

   生成隐藏状态序列：

$$H = \{h_1, h_2, \ldots, h_n\} \tag{14}$$

2. **解码器初始化**
   初始化解码器的隐藏状态 s_0 ，可以用编码器的全局信息初始化。
3. **生成上下文向量**
   对于第 t 步解码，解码器隐藏状态更新为：

$$s_t = \text{Decoder}(s_{t-1}, y_{t-1}) \tag{15}$$

   其中 y_{t-1} 是上一步选中的输出内容。
4. **注意力机制**
   解码器生成注意力权重 α_t：

$$e_{t,i} = v^\top \tanh(W_h h_i + W_s s_t + b) \tag{16}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j \notin Y_{1:t-1}} \exp(e_{t,j})}, \quad i \in \{1, 2, \ldots, n\} \tag{17}$$

   其中：
   - v, W_h, W_s, b 是可学习的参数

- $e_{t,i}$ 表示当前状态 s_t 与候选项 x_i 的相关性
- 为避免重复选择，softmax 分母只包括尚未被选中的索引

5. **输出索引选择**
   根据注意力权重选择输出项索引：

$$y_t = \arg\max_i \alpha_{t,i} \tag{18}$$

6. **终止条件**
   当输出序列长度达到预定值 L 或其他终止条件满足时，停止解码。

GRN

GRN（Generative Rerank Network）采用评估器（Evaluator）和生成器（Generator）实现上下文感知重排序框架。

评估器（Evaluator）

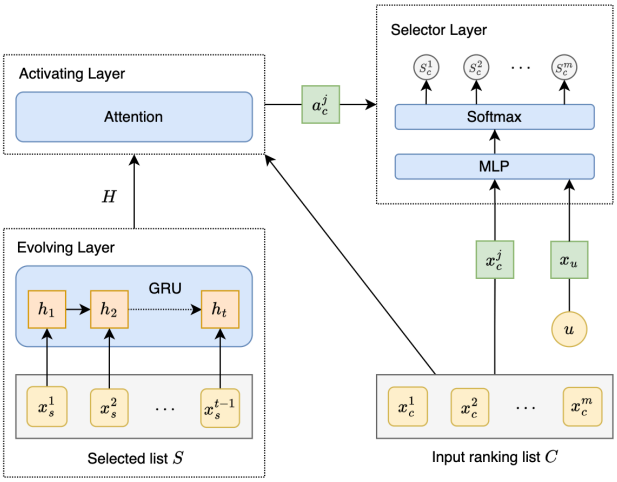
评估器采用双向 LSTM 和自注意力机制来建模标注的最终排序列表中的上下文信息，其目标是更精确地预测每个文档/物品在最终排序列表中的上下文交互概率：

$$E\left(x_v^t \mid u, \mathcal{V}; \Theta^E\right) = \sigma\left(f\left(f\left(f\left(x_u \oplus x_v^t \oplus h_t \oplus a_t\right)\right)\right)\right) \tag{19}$$

其中，x_u 和 x_v^t 分别为用户和最终排名列表 $\mathcal{V}$ 中第 t 个物品的向量表示，h_t 是 LSTM 的输出状态，a_t 是自注意力机制的输出，$f(x) = \mathrm{ReLU}(Wx + b)$，$\sigma(\cdot)$ 是逻辑函数。评估器的参数集为 $\Theta^E = \{W, b\}$，即 Bi-LSTM 和 MLP 的参数联合。

生成器（Generator）

生成器，结合GRU、注意力机制和指针网络来逐步选择输入排序列表中的物品。生成器的目标是学习一个重排序策略，以生成最终的排序列表。生成器包含演化层、激活层、选择层：



- **演化层（Evolving Layer）**
  - 用户的意图或兴趣会随着时间的推移而变化，演化层采用 GRU 模块，在每一步将序列信息提炼为目标用户的状态表示，以学习目标用户在浏览历史中的动态表示。在第 t 步，设 $\mathcal{S} = \left[x_s^0, x_s^1, \ldots, x_s^{t-1}\right]$ 是从输入排名列表 C 中选出的 $t-1$ 个项目，GRU 模块的输出可以计算如下：

$$
\begin{aligned}
z_t &= \sigma\left(W_z x_l^{t-1} + U_z h_{t-1}\right) \\
r_t &= \sigma\left(W_t x_l^{t-1} + U_t h_{t-1}\right) \\
\widetilde{h}_t &= \tanh\left(W x_l^{t-1} + U\left(r_t h_{t-1}\right)\right) \\
h_t &= (1 - z_t) h_{t-1} + z_t \widetilde{h}_t
\end{aligned} \tag{20}
$$

- 其中W_z, U_z, W_t, U_t, W和 U 是可训练的变量，$H = [h_1, \ldots, h_t]$ 表示选定列表 $\mathcal{S}$ 的序列编码。

- **激活层（Activating Layer）**
  - 激活层在选定的项目的顺序表示条件下，使用注意力机制来计算每个剩余项目的重要性权重。给定输入排名列表中第 j 个剩余项目的表示 x_c^j 和已选择列表的序列表示 H，采用加权和生成输入排名列表中第 j 个项目的新表示，相应的注意力权重如下：

$$
\begin{aligned}
a_h^i &= \frac{\exp\left(h_i W x_c^j\right)}{\sum_{i=1}^{t} \exp\left(h_i W x_c^j\right)} \\
a_c^j &= \sum_{i=1}^{t} a_h^i h_i
\end{aligned} \tag{21}
$$

- **选择层（Selector Layer）**
  - 在获得输入排名列表中项目的表示 a_c^j 之后，采用指针网络选择最合适的项目加入到最终排名列表中。对于输入排名列表 C 中的第 j 个项目 x_c^j，在第 t 步采用 MLP 实现特征交互：

$$
\tilde{s}_c^j = f\left(f\left(f\left(x_u \oplus x_c^j \oplus a_c^j\right)\right)\right) \tag{22}
$$

  - 之后采用 softmax 函数将向量 $\tilde{s}_c^j$ 归一化：

$$
s_c^j = \frac{\exp\left(\tilde{s}_c^j\right)}{\sum_{j}^{m} \exp\left(\tilde{s}_c^j\right)} \tag{23}
$$

  - 其中 m 为 C 中项目的数量。随后在第 t 步，生成器将选择具有最高选择概率的项目（排除已选中的项目）：

$$
G^t\left(u, C; \Theta^G\right) = x_c^{\arg\max_j s_c^j} \quad \text{s.t.} x_c^{\arg\max_j s_c^j} \notin \mathcal{S} \tag{24}
$$

优势奖励

GRN 采用交叉熵损失函数训练评估器，并使用策略梯度和优势奖励训练生成器。优势奖励包括自我奖励和差分奖励：

$$
r\left(x_o^t \mid u, O\right) = r^{\text{self}}\left(x_o^t \mid u, O\right) + r^{\text{diff}}\left(x_o^t \mid u, O\right) \tag{25}
$$

其中，r^{self} 是自我奖励，r^{diff} 是差分奖励，x_o^t 为生成的最终排名列表 $O = \left[x_o^1, \ldots, x_o^n\right]$ 中第 t 个项目。

1. **Self reward（自奖励）**：
   - 自奖励指项目在最终排名列表中自身的交互概率所带来的奖励，奖励由评估器（Evaluator）预测：

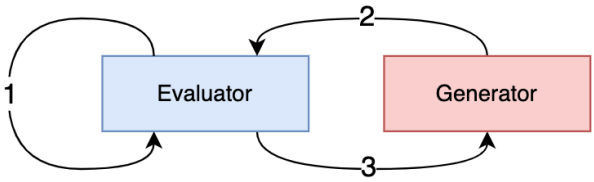
$$
r^{\text{self}}\left(x_o^t \mid u, O\right) = E\left(x_o^t \mid u, O; \Theta^E\right) \tag{26}
$$

2. **Differential reward（差分奖励）**：
    - 差分奖励指一个项目在最终排名列表中对其他项目交互概率的影响所带来的额外奖励，即使其自身的交互概率较低，但如果有助于提升后续项目的选择概率，仍应给予正向奖励。差分奖励通过比较包含某个项目时和其他项目交互概率的总和与不包含该项时的概率总和之间的差异来计算

$$r^{\text{diff}}\left(x_o^t \mid u, O\right) = \sum_{x_o^i \in O^-} E\left(x_o^i \mid u, O; \Theta^E\right) - \sum_{x_o^i \in O^-} E\left(x_o^i \mid u, O^-; \Theta^E\right) \tag{27}$$

    - 其中，O^- 是 O 中不包含 x_o^t 的集合。

训练流程



训练过程如下步骤迭代，直到生成器的参数收敛：

1. 使用列表交互记录来训练评估器直到参数收敛
2. 生成器为评估器生成最终的排名列表
3. 通过基于评估器的优势奖励的策略梯度更新生成器参数

总结

搜索重排是信息检索中不可或缺的环节，其目标是在精排阶段生成的候选列表基础上，通过更复杂的模型和策略，进一步优化列表的质量，平衡效率与多样性并使业务价值最大化。

重排所面临的挑战有：

1. **上下文一致性问题：**
   打分时的上下文与最终展现的上下文不一致，影响模型效用
2. **多目标优化冲突：**
   点击率、转化率、多样性、用户满意度等目标间存在冲突，难以找到最佳平衡点
3. **全局最优性难以实现：**
   贪心策略无法考虑后续决策对当前排序的反作用，导致全局优化受限
4. **计算效率与在线部署：**
   重排序算法复杂度较高，需要在高效计算与实时响应间找到平衡
5. **用户需求的多样化：**
   不同用户在不同场景下的需求千差万别，单一策略难以覆盖

在上述挑战下，重排可以从以下几个方向发展：

1. **上下文感知的深度建模**

    - 利用 Transformer 等模型构建全局上下文，动态调整排序决策
    - 考虑候选内容的历史交互、用户偏好与序列特征，提升个性化排序效果
2. **生成式重排的广泛应用**
    - 加强序列生成-序列评估的联动，确保生成序列的全局最优性
    - 引入强化学习增强生成策略，提升长期用户满意度
3. **多目标协同优化**
    - 建立统一的多目标优化框架，综合短期和长期目标进行排序
    - 动态调整权重，实现点击率与多样性、内容公平性间的有效平衡
4. **高效模型设计与部署**
    - 利用模型压缩、知识蒸馏等技术降低计算成本
    - 探索轻量化、高效的模型架构以适应实时应用场景
5. **用户互动与反馈机制**
    - 引入用户显性或隐性反馈，构建交互式重排序优化框架
    - 动态调整排序决策以适应用户需求的变化
6. **长期生态建设**
    - 优化长尾内容的曝光，激励内容创作者积极参与
    - 综合考虑用户体验与平台生态，促进系统长期健康发展

参考文献

1. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries
2. Fast Greedy MAP Inference for Determinantal Point Process to Improve Recommendation Diversity
3. Determinantal point processes for machine learning
4. k-DPPs: Fixed-size determinantal point processes
5. Globally Optimized Mutual Influence Aware Ranking in E-Commerce Search
6. Seq2Slate: Re-ranking and Slate Optimization with RNNs
7. Personalized Re-ranking for Recommendation
8. GRN: Generative Rerank Network for Context-wise Recommendation