搜索系统算法解析-OAA

相关性

搜索相关性是搜索引擎的核心，它决定了用户查询Query和系统返回结果Doc之间的匹配度。这种匹配度的高低，对于搜索用户体验有着至关重要的影响。

相关性标准

做好相关性需要一个标准分档体系用来评估和度量 **查询Query** 与 **文档Doc** 之间匹配程度。相关性标准的制定可以帮助搜索系统量化相关性，从而在排序和优化过程中提供依据。不同的搜索产品可能有不同的相关性标准和分档方式，但一般都会基于某些通用的标准化方法。

工业界普遍采用分级标准来衡量查询与文档的相关性。常见的相关性等级有以下几种：

- **0**：**完全不相关（Not Relevant）**
  - 定义：文档与查询没有任何关联
  - 说明：完全无法满足用户的主需或次需。文档内容与用户查询毫无关联，对需求没有任何帮助
  - 示例：
    - 查询：**“iPhone 15 Pro Max 价格”**
      完全不相关文档：讲述 Android 系统发展历程，或完全无关的其他内容
- **1**：**略微相关（Marginally Relevant）**
  - 定义：文档与查询在某些方面有关，但对用户问题的解答效果很差，有一定参考价值
  - 说明：对用户需求几乎没有帮助。用户主需和次需几乎没有覆盖，仅包含与查询主题或意图相关的少量无关紧要的信息。文档主题与查询存在部分语义或背景的弱相关性，但不直接匹配
  - 示例：
    - 查询：**“iPhone 15 Pro Max 价格”**
      略微相关文档：讨论 iPhone 的历史发展或者旧款 iPhone 的价格信息，与当前查询需求相关性较弱
- **2**：**部分相关（Partially Relevant）**
  - 定义：文档与查询有一定的相关性，可以部分回答用户的问题或满足查询需求
  - 说明：文档可能涉及与查询相似或相关的主题，但无法直接满足查询的核心需求。即部分满足用户主需，但信息不全面、不精确或次需偏离
  - 示例：
    - 查询：**“iPhone 15 Pro Max 价格”**
      部分相关文档：讨论 iPhone 15 系列的发布信息，但没有列出价格，或者文档内容中只有一句提到价格信息
- **3**：**高度相关（Highly Relevant）**
  - 定义：文档与查询紧密相关，可以完全满足用户查询的需求

    - 说明：查询的主需求得到清晰且全面的回答，但附加需求（如果有）可能没有涵盖，或者覆盖得较为浅显。文档主题与查询一致，内容表达与用户预期匹配，但可能有次要信息干扰或表达上不够紧凑。即文档的部分次级信息可能稍微偏离查询主题，但总体与查询主旨一致
    - 示例：
        - 查询："iPhone 15 Pro Max 价格"
          高度相关文档：列出 iPhone 15 的价格，但没有明确区分 Pro Max 版本，或仅涉及部分价格信息
- **4**：**完全相关（Perfectly Relevant）**
    - 定义：文档与查询完全匹配，解决了查询提出的所有问题，符合用户期望
    - 说明：查询背后的主需求被文档内容精准覆盖，提供了明确且详尽的答案。对于存在次需的场景，文档的主信息部分集中覆盖主需，次需部分如果涉及则为附加但不是关键点。文档主题与查询主题高度一致，语义、表达方式和查询意图对齐
    - 示例：
        - 查询："iPhone 15 Pro Max 价格"
          完全相关文档：详细列出 iPhone 15 Pro Max 各个版本的价格，清晰明确

如上，文档主体内容是否满足用户主需始终是评估相关性的首要标准，次需是考虑文档优劣的附加因素。若查询语义不明确，以主流用户群体的主要需求为评估标准，结合实际业务目标判断。

相关性技术

搜索相关性技术通过计算 查询Query 和 文档Doc 的相关程度来判断 Doc 是否满足用户 查询Query 的需求。目前主流的相关性技术是采用预训练语义模型的（BERT），除此之外，传统的文本匹配和 GBDT 模型依然可以在较少算力的支持下提供有效信息。

文本匹配

传统的搜索相关性技术依赖于词频、词项匹配等方法，这些方法简单高效，适用于早期的信息检索系统。

常见的相似度匹配算法有 TF-IDF 和 BM25，其基本思想是，Query 中的词在 Doc 中出现次数越多，则 Query 和 Doc 越相关。

TF-IDF

TF-IDF（Term Frequency-Inverse Document Frequency）用于衡量一个词在文档中的重要性，其结合了词在文档中的频率（TF）和词在整个语料库中的稀有性（IDF）来作为判断依据。

TF-IDF的计算公式为：

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{1}$$

其中，t为目标词项（term），d为目标文档，D为语料库（包含所有文档）。

- **词频（TF）**

$$\text{TF}(t, d) = \frac{f(t, d)}{\sum_{t' \in d} f(t', d)} \tag{2}$$

    - $f(t, d)$：词 t 在文档 d 中出现的次数

- $\sum_{t' \in d} f(t', d)$：文档 d 中所有词出现的总次数
- TF 是一个局部指标，表示一个词在当前文档中的重要性。词在文档中出现得越多，其重要性可能越高

- **逆文档频率（IDF）**

$$\mathrm{IDF}(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|} \tag{3}$$

  - $||D||$：语料库中文档的总数
  - $||d \in D : t \in d||$：包含词 t 的文档数量（文档频率）
  - IDF 是一个全局指标，表示词在整个语料库中的稀有性。词在整个语料库中出现得越少，其区分性越强

通过将 TF 和 IDF 相乘，TF-IDF 能有效提升那些对当前文档具有区分性的关键词的权重，降低全局常见但区分性较弱词的权重。

BM25

BM25 对 TF-IDF 进行了改进，BM25的计算公式为：

$$\mathrm{BM25}(t, d, D) = \mathrm{IDF}_{\mathrm{BM25}}(t, D) \times \mathrm{TF}_{\mathrm{BM25}}(t, d, D) \tag{4}$$

其中，t为目标词项（term），d为目标文档，D为语料库（包含所有文档）。

- **词频（TF修正）**

$$\mathrm{TF}_{\mathrm{BM25}}(t, d, D) = \frac{f(t, d) \cdot (k_1 + 1)}{f(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)} \tag{5}$$

  - $f(t, d)$：词 t 在文档 d 中的词频（Term Frequency）
  - $||d||$：文档 d 的长度（即词的总数）
  - avgdl：语料库中文档的平均长度
  - k_1：调节词频饱和程度的参数，通常取值范围为 [1.2, 2]
  - b：调节文档长度归一化的参数，通常取值范围为 [0, 1]
  - 词频饱和：对 $f(t, d)$ 引入非线性加权，防止高词频对得分过度影响
  - 文档长度归一化：引入参数 b 平衡长文档和短文档

- **逆文档频率（IDF修正）**

$$\mathrm{IDF}_{\mathrm{BM25}}(t, D) = \log \frac{N - n(t, D) + 0.5}{n(t, D) + 0.5} \tag{6}$$

  - N：语料库 D 中的文档总数
  - $n(t, D)$：包含词 t 的文档数
  - 增加了平滑项，避免在极端情况下分母为 0，同时使常见词的权重更低

BM25 的改进体现在对 TF 进行非线性调节（饱和处理）和引入文档长度归一化，同时对 IDF 进行了平滑处理，解决了 TF-IDF 中的高词频偏差和长文档劣势问题。

GBDT 模型

采用 **GBDT（Gradient Boosting Decision Tree）** 模型进行相关性分档是一种常见的技术方案。GBDT 作为一种强大的集成学习方法，能够有效地从多维特征中学习到非线性关系。

GBDT 算法

GBDT（Gradient Boosting Decision Tree） 是一种基于梯度提升（Gradient Boosting）框架的集成学习方法，通过将多个弱分类器（通常是决策树）结合起来，逐步优化模型的预测结果。

GBDT 是一种 **加法模型**，每个新加入的决策树（基学习器）通过拟合前一轮模型的残差来修正预测结果。给定一个损失函数和模型的当前预测值，目标是通过训练下一棵树来减少损失函数的值。每棵树在每次迭代中都要拟合 **负梯度**，即残差的负方向：

$$F(x) = F^{(0)} + \sum_{t=1}^{T} \gamma_t h_t(x) \tag{7}$$

其中：$F^{(0)}$ 是初始模型，γ_t 是每棵树的学习率（缩放因子），用于控制每棵树对最终模型的贡献大小，$h_t(x)$ 是第 t 轮训练得到的决策树，T 是树的总数。在每一轮中，新的树 $h_t(x)$ 会学习上一轮预测的残差，学习的过程是基于梯度下降优化目标函数。

GBDT 相关性特征

GBDT以相关性档位为学习目标，输入特征可分为：

- **查询特征**：包括查询的关键词、查询意图等信息
  - **文本特征**：查询的长度、词汇分布、查询类型（精确匹配、模糊匹配等）
  - **行为特征**：用户历史查询、点击模式等
- **文档特征**：包括文档的标题、正文内容、发布时间等信息
  - **文本特征**：文档的长度、主题、内容类型
  - **内容质量**：文档的完整性、关键词覆盖情况、SEO优化情况
- **Query-Doc 特征**：描述查询与文档之间的匹配程度的特征
  - **匹配度**：关键词匹配、词向量相似度等
  - **查询与文档的交互特征**：历史互动行为（点击、评论、分享等）

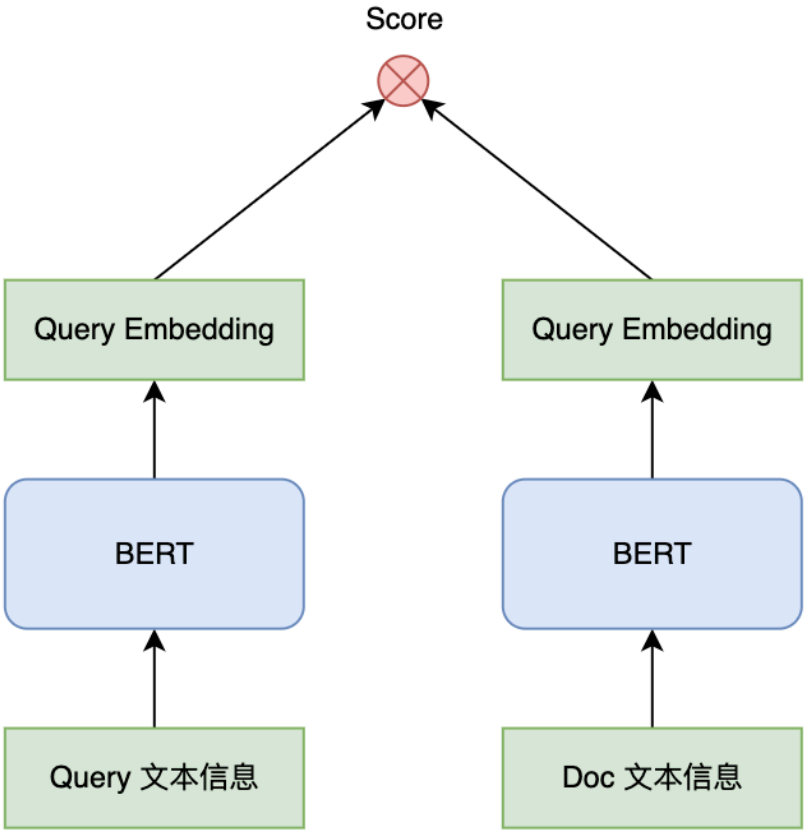
常用的特征如下：

特征
Query 长度/类目/NER/意图/词权重/term 紧密度
Doc 长度/类型/主题/标签/内容类型
Query 和 Doc 向量相似度（Cosine 距离）
Query 和 Doc 标题文本匹配度 （BM25、LCS、Jaccard、SimHash、Levenshtein）
Query 和 Doc 正文文本匹配度 （BM25、LCS、Jaccard、SimHash、Levenshtein）
Query 和 Doc 类目/主题/NER/核心词/标签等匹配度
Query 和 Doc 多域文本匹配度 (OCR/ASR/评论/历史点击 Doc 的 Query)
Query 和 Doc 紧密命中程度
Query 和 Doc 历史互动特征

BERT双塔模型

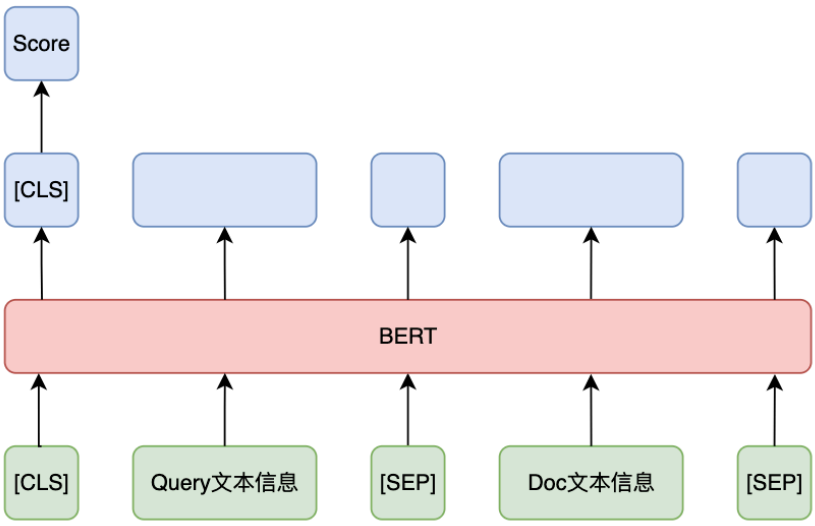
作为搜索排序重要信号，相关性计算贯穿粗排和精排。由于粗排需要计算的候选 Doc 较多，考虑到性能压力，BERT 双塔模型通常被选择用在粗排阶段的相关性计算中。

BERT 的双塔模型分为 Query塔 和 Doc塔，其中 Query塔 的输入通常是 Query文本 加上 Query 的NER、类目等文本特征，Doc塔 的输入一般为文档标题、正文内容、关键字、主题、摘要等文本特征，若文档内容过长，则需要做截断处理或者提取文档摘要作为输入。

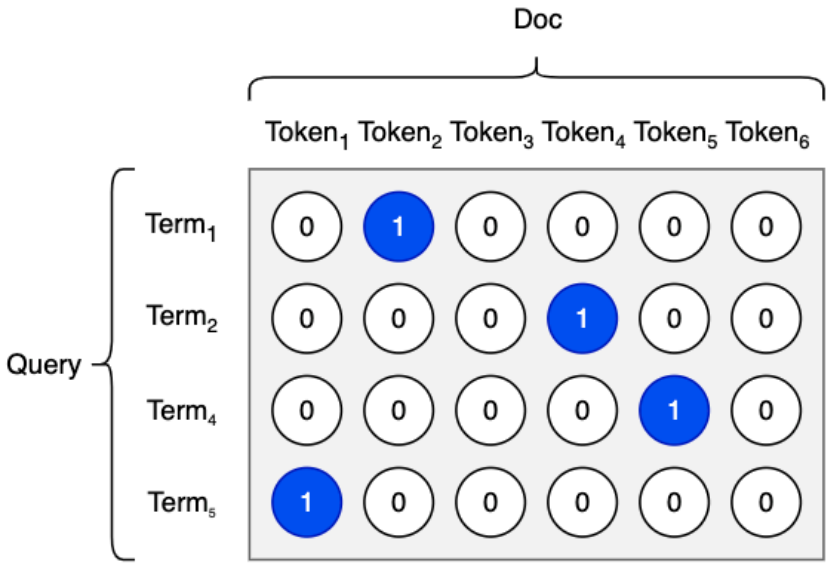


BERT单塔模型

基于交互计算的 BERT单塔 模型能够直接捕捉 Query 和 Doc 之间的细粒度交互信息，由于计算量较大，普遍应用与精排阶段的相关性计算中。



业内通过在模型中引入显示匹配矩阵建模字词的匹配关系，并结合BERT深度语义信息，可以有效带来效果提升。匹配矩阵内元素通过 0 或 1 描述 Query 中的 Term 和 Doc 中的 Token 是否相同：



字词匹配矩阵通过和BERT输出的 Query 和 Doc 向量融合后再做相关性打分计算。

模型训练

多阶段训练

BERT 相关性模型通常需要多阶段的训练，在训练中通过海量的搜索互动数据和人工标注数据，逐步优化模型对 Query 和 Doc 之间语义匹配的理解能力。

1. **通用预训练**
    - 数据：大规模通用语料
    - 任务：
        - MLM：全词掩码，短语掩码，实体掩码，动态掩码，Span-Level 掩码等
        - NSP/SOP：预测下一句，预测排序
2. **领域预训练**
    - 数据：海量搜索点击数据
    - 任务：
        - MLM：全词掩码，短语掩码，实体掩码，动态掩码，Span-Level 掩码等
        - 点击预测：判断 Query 和 Doc 是否发生搜索点击行为
3. **领域微调**
    - 数据：海量搜索点击数据
    - 任务：
        - 文档排序：采用 Pairwise 对同一 Query 下多个候选 Doc 排序
4. **任务微调**
    - 数据：人工/大模型标注数据
    - 任务：

- 相关性分档：
  - 分类任务：对标注档位进行分类预测
  - 回归任务：将档位映射为浮点数，转为回归任务可以捕捉到类别之间的有序性，同时可以缓解不平衡问题

模型蒸馏

BERT 由于其较高的推理耗时，直接部署线上通常会采用知识蒸馏的方式将训练好的大模型知识迁移到小模型中，这种方法通常要比直接训练小模型效果要好。在相关性算法阶段，通常用精排单塔模型蒸馏粗排相关性双塔模型，48层大模型给6层小模型蒸馏。

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：

$$\mathcal{L}_{\text{hard}} = -\sum_{i} y_i \log(\hat{y}_{s,i}) \tag{8}$$

  - 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：

$$\mathcal{L}_{\text{soft}} = D_{\text{KL}}(\hat{y}_t \parallel \hat{y}_s) = \sum_{i} \hat{y}_{t,i} \log\left(\frac{\hat{y}_{t,i}}{\hat{y}_{s,i}}\right) \tag{9}$$

  - 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：

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{hard}} + \beta \cdot \mathcal{L}_{\text{soft}} \tag{10}$$

  - 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：

$$\hat{y}_{t,i} = \frac{e^{z_{t,i}/T}}{\sum_j e^{z_{t,j}/T}} \tag{11}$$

  - 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

样本构建

在训练数据的正负样本构建上，需要结合搜索交互行为和语义属性特征，尽可能的提升训练数据的质量：

- 正样本：
  - 点击特征筛选：选取充足曝光下高CTR样本，筛选 Query 下文档点击率高于平均值的样本
  - 语义属性筛选：根据 Query/Doc 类目/主题动态调整阈值
- 负样本：
  - Skip-Above采样：仅选取在点击文档之前且CTR小于阈值的充分曝光的文档作为负例
  - 随机负采样：同Batch中其他样本做负样本（去除类目一致或文本高度匹配的噪声样本）
  - 数据增强：替换 Query 中核心Term构造负样本
  - 跨类目采样：在同一父类目的不同叶子类目中进行负样本

损失函数

Pointwise Loss

交叉熵（Cross Entropy，CE）

当相关性分档采用**分类任务**时，通常采用交叉熵损失，交叉熵主要用于衡量两个概率分布之间的差异：

$$\mathrm{CE}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i) \tag{12}$$

- 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

均方误差（Mean Squared Error，MSE）

当相关性分档采用**回归任务**时，通常采用 MSE 损失，MSE 通过计算预测值与真实值之间的差异来衡量模型的性能：

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

- n 是样本的总数量
- y_i 是第 i 个样本的真实值（目标值）
- $\hat{y}_i$ 是第 i 个样本的预测值
- $(y_i - \hat{y}_i)$ 是第 i 个样本的误差

Pairwise Loss

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：

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{2}\left[y \cdot d^2 + (1-y) \cdot \max(0, m-d)^2\right] \tag{14}$$

- 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：

$$\mathcal{L}_{\text{RankNet}} = -\sum_{i,j}\left[p_{ij}\log\hat{p}_{ij} + (1-p_{ij})\log(1-\hat{p}_{ij})\right] \tag{15}$$

- 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

相关性评估

相关性常用指标有：

- AUC
  - 对于每个档位，计算该类别与其他类别的区分能力。即当前档位将作为**正类**，其余类别作为**负类**。AUC 值越接近 1，表示搜索相关性算法的排序性能越好，即越能将相关的文档排在前面，不相关的文档排在后面

$$\text{AUC}_{\text{OvR}} = \frac{1}{C}\sum_{i=1}^{C}\text{AUC}_i \tag{16}$$

- DCG
  - 通过计算排序结果中相关文档的累积增益（Cumulative Gain，CG），并根据文档的位置对其增益进行折扣，以便评估排序的质量

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{rel(i)}{\log_2(i+1)} \tag{17}$$

- 其中：
  - DCG_p 表示在前 p 个文档的DCG值
  - $rel(i)$ 表示排名第 i 的文档的相关性评分
  - i 是文档的排名（从1开始，越前面的文档排名越高）
  - $\log_2(i+1)$ 是折扣因子，用来模拟文档排名越靠后的文档，用户查看它的概率越低，因此越低的排名对应着较低的增益

- PNR
  - 在一个排序列表中，计算正序对的数量与逆序对的数量之比。正序对是指在排序中，相关性更高的文档排在相关性较低的文档前面的对数，逆序对则相反。PNR 值越大，说明整个排序列表中正序的比例越多，即搜索结果的排序越合理

总结

搜索相关性衡量搜索引擎返回的结果与用户查询意图的匹配程度，是评判搜索系统质量的一个关键因素。相关性通常和搜索效率指标冲突（如点击率、转化率等），如一些为了吸引用户点击但相关性不强的文档会对用户产生误导诱发点击，当相关性模块将类似文档过滤就有可能带来搜索点击率的下降。

所以相关性必须有精确标准的一套相关性分档体系，帮助搜索系统在各种维度上优化排序结果。即，相关性策略的目标需要平衡相关性和搜索效率，清楚自己的功能定位，约束搜索排序以避免过度追求点击率而忽视了用户满意度，从长远上提高搜索系统的整体质量和用户体验。