12 | 非精准Top K检索：如何给检索结果的排序过程装上“加速器”？

2020-04-22 陈东

检索技术核心20讲 进入课程 >

讲述：陈东
时长 17:47 大小 16.30M

你好，我是陈东。

上一讲，我们详细讲解了 Top K 检索的打分排序过程，并且还提到可以使用堆排序代替全排序，来大幅降低排序的时间代价。然而，对于这整个检索过程来说，精准复杂的打分开销要比排序大得多。因此，如果我们想更大幅度地提升检索性能，优化打分过程是一个重要的研究方向。那打分过程具体该怎么优化呢？今天，我们就来聊聊这个问题。

什么是非精准的 Top K 检索？

想要优化打分过程，一个很自然的思路就是通过简化打分机制，来降低打分开销。但是简化之后，我们的排序结果就不精准了。这该怎么办呢？这个问题先不着急解决，我们先来看看不精准的排序结果对用户会有什么影响。

其实，在搜索引擎中，排在第一页的结果并不一定是分数最高的。但由于用户在搜索时，本来就没有明确的目标网页，所以只要第一页的网页内容能满足用户的需求，那这就是高质量的检索结果了。

不仅如此，在推荐引擎中也是一样。推荐系统会根据用户的历史行为进行推荐，可推荐的物品非常多。比如说，如果用户曾经购买过《C++ 程序设计》这本书，那接下来我们既可以推荐《C++ 编程实战》，也可以推荐《C++ 编程宝典》。无论我们推荐哪一本，可能对用户来说差别都不大。

我们发现，其实在很多实际的应用场景中，**高质量的检索结果并不一定要非常精准，我们只需要保证质量足够高的结果，被包含在最终的 Top K 个结果中就够了**。这就是**非精准 Top K 检索的思路**。

实际上，在工业界中，我们会使用非精准 Top K 检索结合精准 Top K 检索的方案，来保证高效地检索出高质量的 结果。具体来说，就是把检索排序过程分为两个阶段：第一阶段，我们会进行非精准的 Top K 检索，将所有的检索结果进行简单的初步筛选，留下 k1 个结果，这样处理代价会小很多（这个阶段也被称为召回阶段）；第二个阶段，就是使用精准 Top K 检索，也就是使用复杂的打分机制，来对这 k1 个结果进行打分和排序，最终选出 k2 个最精准的结果返回（这个阶段也被称为排序阶段）。

其实，这个流程你应该很熟悉。这就像我们在招聘时，会先根据简历筛选，再根据面试结果进行筛选。简历筛选的效率很高，但是不精准；面试比较耗时，但能更好地判断候选人的能力，这就属于精准挑选了。

再说回到工业界的检索方案，非精准 Top K 检索到底是怎么使用简单的机制，来“加速”检索过程的呢？加速的效果如何呢？我们一起来看看。

非精准 Top K 检索如何实现？

在非精准 Top K 检索中，一个降低打分计算复杂度的重要思路是：**尽可能地将计算放到离线环节，而不是在线环节**。这样，在线环节我们就只需要进行简单的计算，然后快速截断就

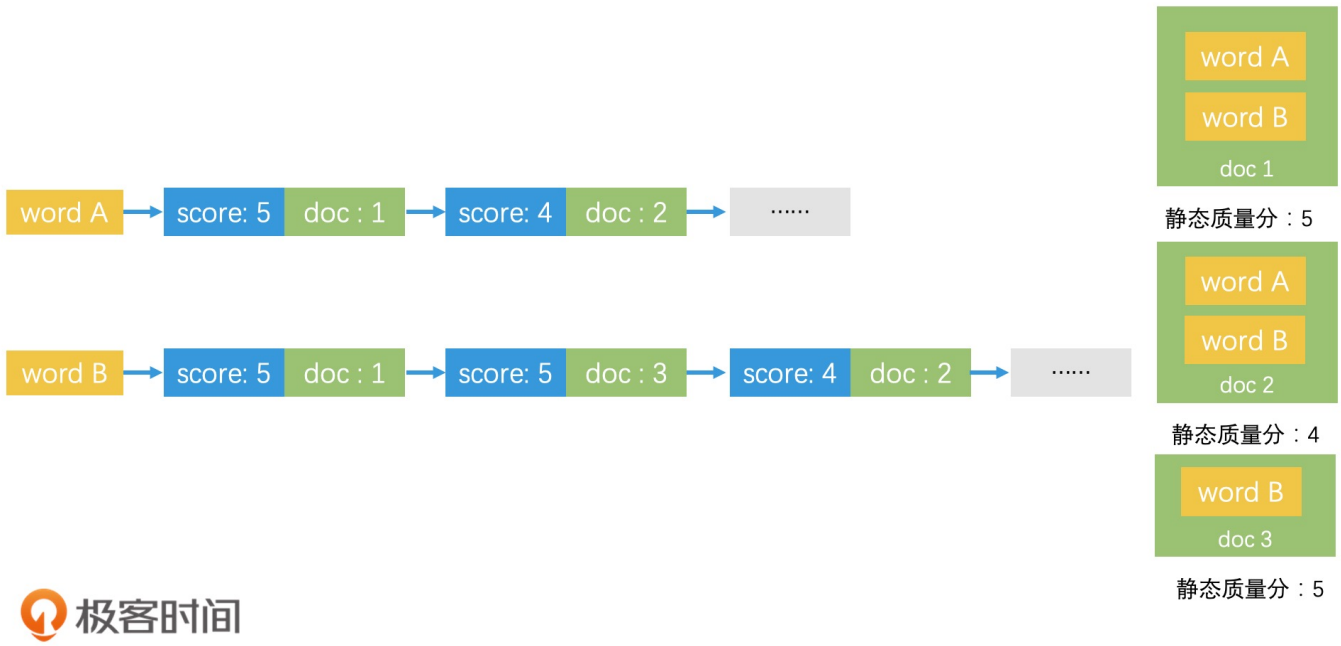
可以了。一个极端的方案就是根据检索结果的静态质量得分进行打分和截断。具体该怎么做呢？我们一起来看。

1. 根据静态质量得分排序截断

所谓静态质量得分，指的是不考虑检索结果和实时检索词的相关性，打分计算仅和结果自身的质量有关。这样，所有的打分计算就都可以在离线环节完成了。也就是说，我们只需要根据离线算好的静态质量得分直接截断，就可以加速检索的过程了。这么说可能比较抽象，我们通过一个例子来解释一下。

以搜索引擎为例，我们可以不考虑搜索词和网页之间复杂的相关性计算，只根据网站自身的质量打分排序。比如说，使用 Page Rank 算法（Google 的核心算法，通过分析网页链接的引用关系来判断网页的质量）离线计算好每个网站的质量分，当一个搜索词要返回多个网站时，我们只需要根据网站质量分排序，将质量最好的 Top K 个网站返回即可。

不过，为了能快速返回 Top K 个结果，我们需要改造一下倒排索引中的 posting list 的组织方式。我们讲过，倒排索引的 posting list 都是按文档 ID 进行排序的。如果希望根据静态质量得分快速截断的话，那我们就应该将 posting list 按照静态质量得分，由高到低排序。对于分数相同的文档，再以文档 ID 二次排序。



按静态质量得分排序

这样一来，在检索的时候，如果只有一个关键词，那我们只需要查出该关键词对应的 posting list，截取前 k 个结果即可。但是如果我们要同时查询两个关键词，截断的过程就会复杂一些。尽管比较复杂，我们可以总结为两步：第一步，我们取出这两个关键词的 posting list，但不直接截断；第二步，我们对这两个 posting list 归并排序。留下分数和文档 ID 都相同的条目作为结果集合，当结果集合中的条目达到 k 个时，我们就直接结束归并。如果是查询多个关键词，步骤也一样。

那在这个过程中，我们为什么可以对这两个 posting list 进行归并排序呢？这是因为文档是严格按照静态质量得分排列的。如果文档 1 的分数大于文档 2，那在这两个 posting list 中文档 1 都会排在文档 2 前面。而且，对于分数相同的文档，它们也会按照 ID 进行二次排序。所以，任意的两个文档在不同的 posting list 中，是会具有相同的排序次序的。也因此，我们可以使用归并的方式来处理这两个 posting list。

总结来说，在使用静态质量得分选取非精准 Top K 个结果的过程中，因为没有实时的复杂运算，仅有简单的截断操作，所以它和复杂的精准检索打分相比，开销几乎可以忽略不计。因此，在对相关性要求不高的场景下，如果使用静态质量得分可以满足系统需求，这会是一个非常合适的方案。但如果应用场景对相关性的要求比较高，那我们还得采用其他考虑相关性的非精准检索方案。

2. 根据词频得分排序截断

既然说到了相关性，就必须要提到词频了。我们在上一讲说过，词频记录了一个关键词在文档中出现的次数，可以代表关键词在文档中的重要性。而且，词频的计算是在索引构建的时候，也就是在离线环节完成的，并且它还可以直接存储在 posting list 中。

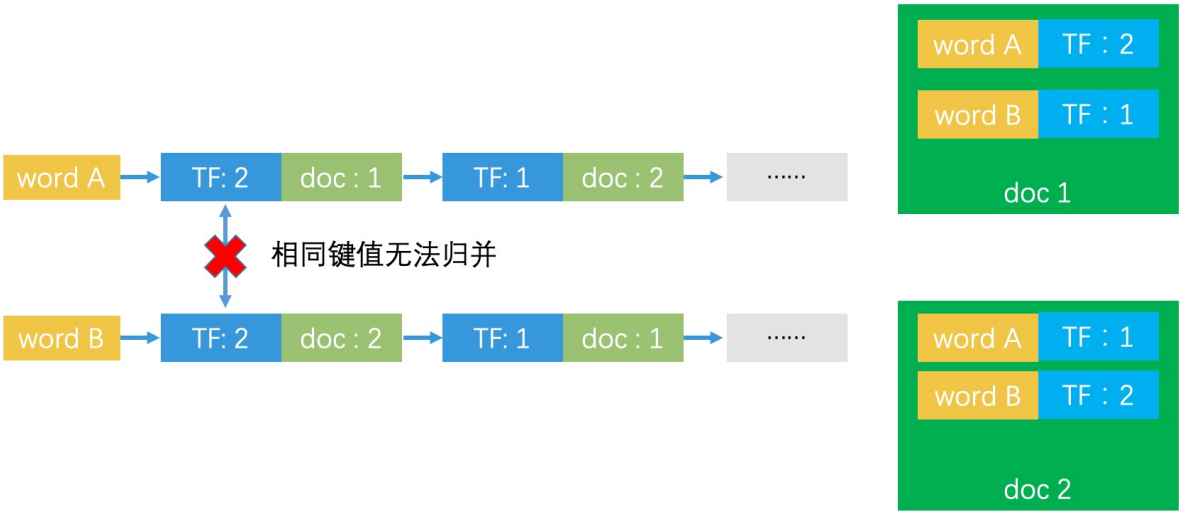
这就给了我们一个启发，我们可以考虑使用词频来对 posting list 中的文档进行截断。具体该怎么做呢？我们可以像使用静态质量得分一样，直接使用词频的值降序排序 posting list 吗？你可以先自己想一想，然后和我一起分析。

假设，搜索词中只有一个关键词，那我们只需要查出该关键词对应的 posting list，截取前 k 个结果就可以了。这时候，这个方法是可以正常工作的。

但是如果搜索词中有两个关键词 A 和 B，就可能出现这么一种情况：文档 1 中出现了 2 次关键词 A，1 次关键词 B；文档 2 中出现了 1 次关键词 A，2 次关键词 B。那么，在关键词 A 的 posting list 中，文档 1 的分数是 2，文档 2 的分数是 1，文档 1 排在文档 2 前面。

但是在关键词 B 的 posting list 中，文档 2 的分数是 2，文档 1 的分数是 1，文档 2 排在文档 1 前面。

这个时候，文档 1 和文档 2 在不同的 posting list 中的排序是不同的，因此，我们无法使用归并排序的方法将它们快速合并和截断。



极客时间

以词频数值排序导致无法归并

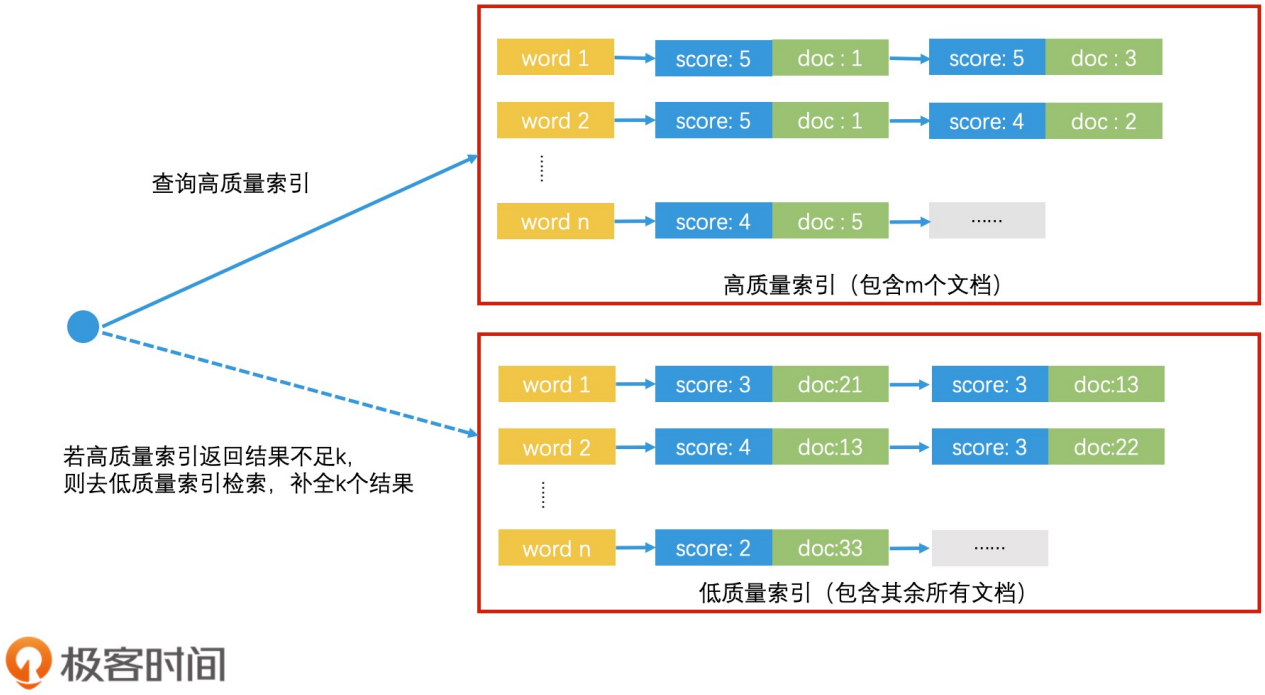
既然问题出在排序上，那我们能否既用上词频的分值，又保持 ID 有序呢？有这么一个解决思路，就是对 posting list，我们先根据词频大小选出远多于 k 的前 r 个结果，然后将这 r 个结果按文档 ID 排序，这样就兼顾了相关性和快速归并截断的问题。这种根据某种权重将 posting list 中的元素进行排序，并提前截取 r 个最优结果的方案，就叫作**胜者表**。

胜者表的优点在于，它的排序方案更加灵活。比如说，我可以同时结合词频和静态质量得分进行排序（比如说权重 = 词频 + 静态质量得分），这样就同时考虑了相关性和结果质量两个维度。然后，我们对于每个 posting list 提前截断 r 个结果，再按文档 ID 排序即可。

但是有一点需要注意，胜者表的提前截断是有风险的，它可能会造成归并后的结果不满 k 个。比如说，文档 1 同时包含关键词 A 和 B，但它既不在关键词 A 的前 r 个结果中，也不在关键词 B 的前 r 个结果中，那它就不会被选出来。在极端情况下，比如，关键词 A 的前 r 个结果都是仅包含 A 的文档，而关键词 B 的前 r 个结果都是仅包含 B 的文档，那关键词 A 和 B 的前 r 个结果的归并结果就是空的！这就会造成检索结果的丢失。

3. 使用分层索引

对于胜者表可能丢失检索结果的问题，我们有一种更通用的解决方案：**分层索引**。我们可以同时考虑相关性和结果质量，用离线计算的方式先给所有文档完成打分，然后将得分最高的 m 个文档作为高分文档，单独建立一个高质量索引，其他的文档则作为低质量索引。高质量索引和低质量索引的 posting list 都可以根据静态质量得分来排序，以方便检索的时候能快速截断。那具体是怎么检索的呢？我们一起来看看。



分层索引的Top K检索

在实际检索的时候，我们会先去高质量索引中查询，如果高质量索引中可以返回的结果大于 k 个，我们直接截取 Top K 个结果返回即可；如果高质量索引中的检索结果不足 k 个，那我们再去低质量索引中查询，补全到 k 个结果，然后终止查询。通过这样的分层索引，我们就能快速地完成 Top K 的检索了。

相比于前面两种优化方案，分层索引是最通用的一种。而且，分层索引还可以看作是一种特殊的索引拆分，它可以和我们前面学过的索引拆分技术并存。比如说，对于高质量索引和低质量索引，我们还可以通过文档拆分的方式，将它们分为多个索引分片，使用分布式技术来进一步加速检索。

到这里，非精准 Top K 检索的三种实现方法我们都讲完了。总结来说，这些方法都是把非精准 Top K 检索应用在了离线环节，实际上，非精准 Top K 检索的思想还可以拓展应用到

在线环节。也就是说，**我们还能在倒排检索结束后，精准打分排序前，插入一个“非精准打分”环节**，让我们能以较低的代价，快速过滤掉大部分的低质量结果，从而降低最终进行精准打分的性能开销。

除此之外，我还想补充一点。我们说的“非精准打分”和“精准打分”其实是相对的。这怎么理解呢？

举个例子，如果我们的“精准打分”环节采用的是传统的机器学习打分方式，如逻辑回归、梯度下降树等。那“非精准打分”环节就可以采用相对轻量级的打分方案，比如说采用 TF-IDF 方案，甚至是 BM25 方案等。而如果“精准打分”环节采用的是更复杂的深度学习的打分方式，比如使用了 DNN 模型，那么相对来说，“非精准打分”环节就可以采用逻辑回归这些方案了。

所以说，无论非精准打分的方案是什么，只要和精准打分相比，“能使用更小的代价，快速减少检索范围”，这就足够了。而这也是在前面多次出现过的检索加速的核心思想。

重点回顾

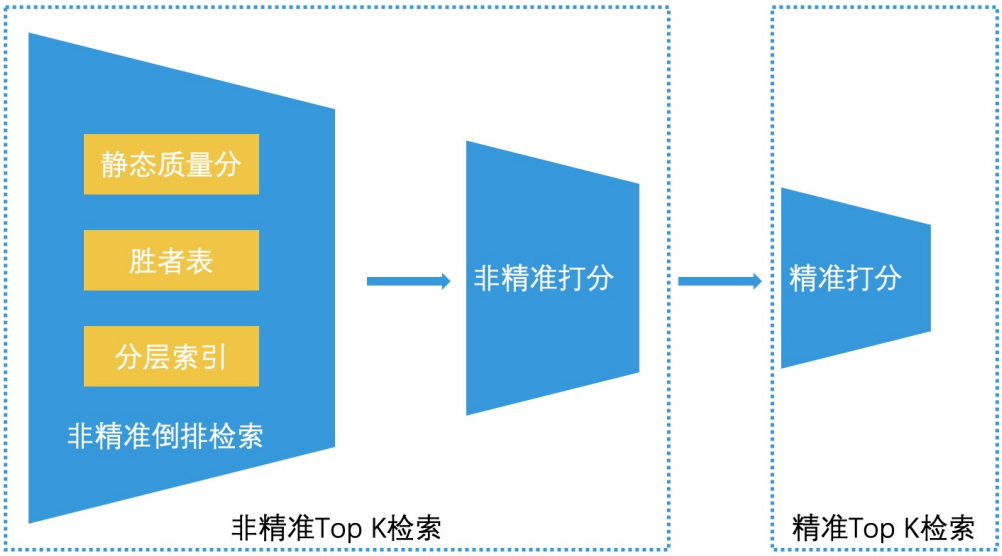
今天，我们主要学习了利用非精准 Top K 检索为检索过程“加速”。

非精准 Top K 检索实现加速的方法主要有三种，分别是根据静态质量得分排序截断，以及使用胜者表，利用词频进行相关性判断进行截断，还有使用分层索引，对一次查询请求进行两层检索。

这三种方法的核心思路都是，尽可能地将计算从在线环节转移到离线环节，让我们在在线环节中，也就是在倒排检索的时候，只需要进行少量的判断，就能快速截断 Top K 个结果，从而大幅提升检索引擎的检索效率。

此外，我们还能将非精准 Top K 检索拓展到线上环节，通过引入“非精准打分”的环节，来进一步减少参与“精准打分”的检索结果数量。

最后，在工业界中，完整的 Top K 检索是由非精准 Top K 检索和精准 Top K 共同完成的。这种设计的核心思想，是希望用更小的代价快速减少检索排序范围，从而提升整体在线检索的效率。我把它的实现过程总结成了一张示意图，你可以参考它来梳理、巩固今天的内容。



极客时间

完整Top K检索的过程示意图

课堂讨论

1. 在分层索引中，posting list 中的文档为什么还要根据静态质量得分排序？排序应该是升序还是降序？
2. 对于非精准 Top K 检索，你有没有相关的方法或者应用场景可以分享呢？

欢迎在留言区畅所欲言，说出你的思考过程和最终答案。如果有收获，也欢迎把这一讲分享给你的朋友。

上一篇　11 | 精准Top K检索：搜索结果是怎么进行打分排序的?

精选留言 (6)

写留言

一步
2020-04-22

看完这篇文章和上一篇文章：我的理解是
静态质量得分的非精确打分和精确打分的打分对象是不同的
静态质量得分非精确打分因为是离线打分，针对的是文档进行打分，该文档的所有关键词的分都一样
精确打分：针对的某个关键词和文档的相关性进行打分

展开

作者回复: 是的。静态质量得分不考虑相关性，因此只需要对文档自身进行打分就好。写成函数形式就是f(d)，d代表文档。
而精确打分要考虑查询词和文档的相关性。如果文档是d，查询词是q，那么打分函数就是f(q，d)。由于查询词是动态的，因此只能在实时环节进行打分，无法在离线环节打分。(注意:查询词可能有多个词项，比如同时有“极客”和“时间”两个词项，因此只能实时计算相加。否则如果查询词只有一个词项，那么是可以用胜者表的方法离线计算相关性的)。

一步
2020-04-22

第一步，我们取出这两个关键词的 posting list，但不直接截断；第二步，我们对这两个 posting list 归并排序。留下分数和文档 ID 都相同的条目作为结果集合，当结果集合中的条目达到 k 个时，

这里面为什么也要要求分数也相当呢？ ...
展开

峰
2020-04-22

问题1 降序是必然的，毕竟升序没有意义，但根据静态分降序，然后根据静态分去做粗排，这样不会饿死静态分不高的文档吗，或者它活该饿死。。。。
问题2，没想到什么场景，但思路就是比如可以用一些近似计算的方式算一遍，排除掉一些数据项，再做精确运算。

展开

作者回复: 1.你提了很好的一个问题，静态质量分不好的是否会被活活饿死?其实，不管是静态质量得分还是动态的相关性打分，既然是排序，肯定会有优胜劣汰，只是在大数据的场景下，所谓的劣汰不是完全封杀，而是降低出现的概率。
首先，如果快速截断采取的是静态质量分，那么粗排的时候肯定就是换一种打分机制了，比如说采用bm25进行粗排，或者使用简化版的机器学习(为了保证效率，用在粗排阶段的机器学习模型会比较简单，而不是和用在精排阶段的机器学习模型完全一样)。
另一方面，为了保证结果的多样性，我们应该加入exploit &explore机制，就是保证有小比例的流量可以采用随机的方式选择检索结果，而不是靠打分排序，从而给一些初始打分不高的检索结果证明自己的机会。但如果数据依然不好，那么接下来这些低质量的结果被选出来的概率就会越来越低。
2.在搜索引擎和推荐引擎中，使用tf-idf，bm25，甚至是简化的逻辑回归模型来做非精准top k检索，都是常见的使用方案。此外，就像你说的一样，可以用一些简单算法算一遍，快速排除掉一些数据项。这方面其实也有许多研究和论文，比如说雅虎有一篇wand(weak and)算法的论文，就是讲如何根据相似度的上限来快速截断。
其实，使用非精准检索思路的场景非常多，我文中的简历筛选的过程就是一个例子。下一篇你也会看到另一个例子。

那时刻
2020-04-22

老师，您在胜者表提到静态质量得分和词频两个维度综合考虑来计算文档权重，请问在实际操作中有这样的应用么？

展开

作者回复: 实际中是有的。其实就是一个加权的表达式:
w1*tf + w2*静态质量分。
具体w1和w2的权重，可以看你的应用诉求，手动调整。


那时刻
2020-04-22

1.分层索引采用静态质量得分是大概率只扫描第一层索引就可以得到topk。而采用词频相关性的话，为保证文档齐全大概率需要搜索多层索引。效率比较低。分层索引采用降序排序。

作者回复: 就如你说的，分层索引大概率只扫描第一层就能得到top k。因此，在第一层索引候选集充足的情况下，采用静态质量分降序，能更快进行top k截断，而不需要完整检索完第一层索引，从而达到检索加速的目的。


pedro
2020-04-22

问题1，根据静态质量得分来排序可以有效的筛选出高质量的内容，排序应该是降序，将高质量文档放在前面从而更快的打分。
问题2，没想到深度学习的降维打击，让传统经典的打分算法都沦为到非精准打分的环节了，哎，英雄迟暮。

展开

作者回复: 1.是的。还是为了快速截断。如果在高质量文档的索引中也有足够的候选集，那么其实我们也不需要检索完全部文档，因此对于高质量索引，依然采用根据静态质量降序，是能有加速效果的。
2.的确是这样的，现在的召回环节中，就有用逻辑回归进行截断的，当然，这个环节也有人叫做“粗排”环节。不过历史是相似的，在机器学习出来之前，bm25就是精排；在机器学习出来后，bm25就变成粗排了；同理，在深度学习出来以后，机器学习就可以变成粗排了。说不定未来哪天，深度学习也会变成粗排。