开放搜索

最佳实践

最佳实践

功能篇

分词、匹配、相关性、排序表达式

针对目前若干用户遇到的搜索结果与预期不符合的问题进行统一详细说明，并以此为话题展开说明下opensearch在搜索效果方面的功能和后续一些工作方向。

首先，对于搜索来讲，最常见的有两种做法：

1. 数据库的like查询，可以理解为简单的包含关系；
2. 百度、google等搜索引擎，涉及到分词，将查询词根据语义切分成若干词组term（这个是搜索引擎重难点之一），通过term组合匹配给相应文档进行打分，根据分值排序，并最终返回给用户。

opensearch采用的方式与上述搜索引擎做法基本一致。那这里就有三部分内容会影响搜索效果：**1，分词方式；2，匹配方式；3，相关性算分。**

我们来分别说下这三部分在opensearch上的行为和表现。

接下来，我们详细说明下各个字段的展现效果及适用场景，供大家参考。

分词方式

熟悉各类分词是本篇操作的前提，请务必先查阅 字段类型及分词描述 文档，这里不在复述。

匹配方式

原理

分完词后得到若干term，如何召回文档，就涉及到匹配方式。目前opensearch内部默认支持的是AND，即一篇文档中包含全部的term才能被搜索出来。当然这是对同一关键词而言的，除此之外系统还支持多种匹配方式，如AND、OR、RANK、NOTAND以及()，优先级从高到低为()，ANDNOT，AND，OR，RANK。

举例

关系	用法	含义

	query=title:”苹果 手机”	查询title中包含苹果和手机的文档
AND	query=title:’苹果’ AND cate:’手机’	交集。查询title中包含苹果，且cate包含手机的文档
OR	query=title:’苹果’ OR cate:’手机’	并集。查询title中包含苹果，或cate包含手机的文档
RANK	query=title:’苹果’ RANK cate:’手机’	查询title中包含苹果的文档，如果cate包含手机则可以加分
ANDNOT	query=title:’苹果’ ANDNOT cate:’手机’	查询title中包含苹果，但cate不包含手机的文档

案例

问：我文档中包含“吃饭了”，我搜索“吃饭”、“吃饭了”都能召回，搜索“吃饭了吗”没结果？

答：因为目前opensearch是要求全部的分词结果都匹配才能召回文档，上面的“吗”在文档中没有出现，所以无法召回。

问：我只想查找某些词排在最前面的文档，比如以“肯德基”开头的文档；答：目前不支持位置相关召回。

后续工作

1. 支持用户词典，比如纠错、同义词等，可以根据自己的实际场景挖掘出属于自己的专属词典。

   O2O实体词识别，如query=title:’杭州市文一西路’改写为query=title:’杭州市文一西路’ OR (city:’310010’ AND district:’0012’ AND title:’文一西路’)。

   相关搜索功能。

   热词统计、点击反馈等。

使用技巧

1. 可以通过AND，OR等实现强大的搜索功能。
2. 对于一些filter中的过滤字段，如= 、 ！=的需求，可以尽量使用索引字段来做，可以提高查询性能。点击获取更多优化小技巧。

相关性算分

上面提到的都是跟召回相关的技术，召回文档之后，究竟文档如何排序就涉及到相关性。目前opensearch有

sort子句来支持用户自定义排序。如果不设置sort，则默认为sort=-RANK；sort本身支持多维排序，以及升降序的支持。比如sort=-RANK;+bonus，意思为第一位按照相关性降序排序，相关性分值一样的文档再按照bonus升序排列。这里我们重点描述下RANK的用法，RANK即为opensearch中的相关性设置，主要分为两部分：粗排和精排，分别有对应的控制台中的粗精排表达式配置。

原理

Opensearch相关性算分策略为，取召回的rank_size（目前是100万）个文档按照粗排表达式的定义进行算分；取粗排分最高的N个结果(百级别)按照精排表达式进行算分，并排序；然后根据start与hit的设置取相应结果返回给用户。如果用户获取的结果超过了精排结果数N，则后续按照粗排分数排序结果继续展现。

> 粗排表达式：从上面原理介绍中可以看出粗排对性能（latency）的影响非常大，但同时粗排又非常的重要，否则会出现好的文档无法进入精排而导致文档不能被最终展现。所以粗排要尽量的简单有效，目前opensearch的粗排只支持几个简单的正排字段、静态bm25、时效分等因素。

> 精排表达式：通过粗排表达式筛选出较优质的N个文档进行详细排序，精排表达式中支持复杂的数学计算、逻辑等，并且opensearch提供了丰富的典型场景（如O2O类）的function和feature来满足日常的相关性需求。

同时，系统以内置了多个场景的应用结构和排序表达式，可以供大家参考和使用。

举例

场景	表达式	含义
论坛-粗排	static_bm25()	简略文本分
论坛-精排	text_relevance(title)*3+text_relevance(body) +if(text_relevance(title)>0.07,timeliness(create_timestamp),timeliness(create_timestamp)*0.5) +(topped+special+atan(hits)*0.5+atan(replies))*0.1	文本分 、时效分 、其他属性分
O2O-粗排	sold_score+general_score*2	销量、门店综合分值（离线算好）
O2O-精排	2*sold_score+0.5*reward - 10*distance(lon,lat,u_posx,u_posy) + if ((flags&2) \=\=2, 2, 0)+if(is_open\=\=5,10,0) + special_score	销量、配送速度及准点率 、距离 、是否繁忙、是否在营业时间 、人工干预
小说-粗排	static_bm25()*0.7+hh_hot*0.00003	文本分、热度
小说-精排	pow(min(0.5,max(text_releva	分类相关性、标题相关性、作者

	nce(category),max(text_relevance(title), text_relevance(author)))),2) + general_score*2 + 1.5*(1/(1+pow(2.718281,-((log10(hh_hot)-2)*2-5)))))	相关性 、小说质量 、小说热度
电商-粗排	static_bm25()+general_score*2+timeliness(end_time)	文本分、宝贝综合分值、过期时间
电商-精排	text_relevance(title)*3+text_relevance(category) + general_score*2+boughtScore*2 + tag_match(ctr_query_value,doc_value,mul,sum,false,true) +..	文本相关性、类目相关性 、宝贝人气、卖家分 、ctr预估、特征规则分等

案例

问：精排表达式text_relevance(seller_id)报找不到字段答：text_relevance()只支持TEXT及SWS_TEXT类型，其他不可以。

问：查询报2112错误，是什么问题？答：查询语句（query子句）必须与formula相配合，比如query=default:’keyword’，default中包含title和body字段，而formula指定text_relevance(title)+text_relevance(author)则会报错，因为author在default中不存在。

后续工作

1. 优化错误码，目前很多查询错误都会报1000错误，不利于用户定位问题，且容易造成系统有问题的误导，后续会统一进行梳理。
2. 后续会对电商类场景提供更多样的feature来支持，如有类似的需要请联系我们。
3. 发布截断功能，允许用户指定重要字段，构建索引时考虑该字段的值，降低参与粗排文档数、提高参与精排文档数，即降低查询开销又提高搜索质量，让更多好的文档能够排上来。
4. 相关性ABTest调试界面，会在搜索测试页面增加多个排序表达式的对比界面，将每项表达式的值列出来，方便用户进行排序表达式的调整。

使用技巧

1. 排序表达式的算分是在查询结算对每个文档进行计算的，所以如果跟查询无关的部分的计算可以预先离线计算好，新增一个general_score字段来存放，排序的时候只要使用general_score字段即可，避免大量计算过程，提高查询性能。
2. tag_match的feature允许用户将query中的特征与doc中特征做多维运算，在电商场景下有着非常广泛的用途，有类似的需求的用户可以研究下。

3. opensearch提供了丰富的function和feature，使用得当可以获得非常强大的功能。
4. 相关性有很多部分共同组成，各项之间的权重需要根据搜索排序效果不断进行调整以达到一个用户满意的搜索效果。

ARRAY数组类型的前世今生

本文主要对Array类型的使用场景、数据推送及搜索语法进行系统的介绍，方便大家理解。

什么场景下适合使用ARRAY类型？

Array类型即为数组类型，数组类型即由相同类型的若干个元素组织在一起的数据，期望在搜索的时候对于每一个元素都可以执行单独的查询。比如小说的标签tags，包含“悬疑”、“穿越”、“古典”，希望在搜索“悬疑”的时候能找到该篇小说。

如何推送ARRAY类型的数据？

目前OpenSearch支持多种方式的数据推送方式，那我们就从每个途径来分开阐述如何进行数据推送。

API方式

ARRAY类型需要采用JsonArray的方式来上传数据。如：

```
[{"fields": { "id": "0","int_array": [14,85],"float_array": [14.0,85.0],"string_array": ["abc","xyz"]},"cmd": "ADD"}]
```

具体数据上传接口，请参考 开发指南->V3(标准/高级)API参考手册->应用操作接口->数据处理

SDK方式

这里以php sdk为例，其他sdk做法类似。

```
<?php
require('php_2.0.4/CloudsearchClient.php');
require('php_2.0.4/CloudsearchIndex.php');
require('php_2.0.4/CloudsearchDoc.php');

define('ACCESSKEYID', '您的阿里云AccessKeyId');
define('SECRET', '您的阿里云AccessKeySecert');
define('APP_NAME', '您的应用名称');
define('KEY_TYPE','aliyun'); #固定值

#每个应用具体host值请参考应用管理->应用详情->API入口，打开debug接口方便调试
$client = new CloudsearchClient(
```

```
ACCESSKEYID,
SECRET,
array('host' => 'http://opensearch-cn-hangzhou.aliyuncs.com', 'debug' =>true),
KEY_TYPE
);
$doc = new CloudsearchDoc(APP_NAME, $client);
$json = <<<EOF
[{"fields": { "id": "0","int_array": [14,85],"string_array": ["abc","xyz"]},"cmd": "ADD"}]
EOF;
echo $doc->add($json, '您要推送数据的表名');
echo $client->getRequest(); #打印发送的请求串，前提是CloudsearchClient的debug打开
?>
```

数据源方式

数据源配置允许用户对于数据源数据进行多种格式的解析操作，如果定义了ARRAY类型的字段，可以在该字段上选择MultiValueSpliter插件，定义好多值分隔符，比如上例中的tags，在数据库表中字段内容为：“穿越,悬疑,言情”，那么多值分隔符为英文逗号：“,”，如图所示即可。该插件会自动将数据库中字段转化成为引擎识别的ARRAY类型。

ARRAY类型如何进行检索？能实现怎样的效果？

ARRAY类型的每一个元素都可以单独访问，不管是用在query子句，还是filter子句，如上例中的tags字段（内容为：穿越,悬疑,言情），可以通过query=tags:’穿越’来找到该文档；也可以通过query=title:’步步惊心’ &&filter=tags=”穿越”，来实现标签为“穿越”的名字包含“步步惊心”的小说。同时需要注意一点的是，搜索结果对于Array类型是按照字符串返回的，元素之间使用’\t’分隔，而不是数组。

相关问题

Q: 为什么没有text_array类型，text与string_array有什么区别？
A: text类型（包含text、sws_text、nws_text、mws_text）涉及到分词，本身支持的是模糊搜索，所以没有数组的概念，而string_array指的是每个元素的精确匹配，很可能这里的单个元素本身是由多个词组组成的，但是没关系要求的是全部匹配。

Q: 有没有方法获得array类型的元素个数？
A: 系统提供了fieldlen(array_field)的参数，可以获取元素个数。

中文分词-模糊搜索

模糊搜索是指在用户搜索意图不明确时，搜索引擎将用户的查询（query）与待检索的内容（doc）进行模糊匹配，找出与查询相关的内容。是否相关主要从两个方面衡量：一是query是doc中某些内容的全拼或者简拼；二

是query中内容直接在doc中出现。模糊搜索无法精确理解用户的查询意图，返回的结果中可能包括了一大批用户不想要的信息，所以在使用模糊搜索时一定要结合自己的实际场景，慎重使用。

适用场景

模糊搜索主要用户搜索意图不明确或者数据较少想返回更多查询结果的时候。主要包括以下场景。

拼音搜索：

拼音搜索是指doc中的数据为中文，而希望使用全拼或者简拼进行查询的搜索。比如，文档中的内容为开放搜索，用户希望查询”kaifangsousuo” 或者”kfss”时文档能够被召回。模糊搜索支持这样的功能，而且支持query的形式更丰富。以开放搜索为例，模糊搜索支持的query形式如下（注意是双引号查询）：”kai”、”kaifang”、”sousuo”、 “kaifangsousuo”、”k”、”kf”、”ss”、”kfss”。注意：如果希望搜索内容在doc中是相连的，建议在查询词两边加上双引号。拼音搜索中查询词两边最好都使用双引号，这是因为用户输入的拼音是具有一定的意图的，用户搜索”kfss”（开放搜索）其实是希望这些词是要连在一起的。

前缀搜索：

前缀搜索是指搜索以指定前缀开头的内容的检索（不支持中文前缀匹配），比如手机号码搜索。模糊搜索支持的前缀标识符为’^’，如果用户想搜以138开头的手机号，query可写成”^138”（注意是双引号查询）。不支持中文前缀匹配），比如手机号码搜索。模糊搜索支持的前缀标识符为’^’，如果用户想搜以138开头的手机号，query可写成”^138”（注意是双引号查询）。

后缀搜索：

后缀搜索是指搜索以指定后缀结尾的内容的检索（不支持中文后缀匹配），比如手机号码搜索。模糊搜索支持的后缀标识符为’$’，如果用户想搜以9527结尾的手机号，query可以写成”9527$”（注意是双引号查询）。

单字或单字母搜索：

模糊搜索支持单字或单字母搜索，比如’开放搜索 open search’，通过’放’或者’o’都可以召回。单字或单字母这种使用场景主要是为了扩大召回结果，返回的结果可能不是很准确。

使用与限制

用户创建应用时，将需要进行模糊搜索的字段设置为short_text即可使用模糊搜索。模糊搜索返回的结果默认按照命中的词在字段的前后位置进行排序。比如某个应用的title字段需要模糊搜索，doc1的内容为开放搜索，doc2的内容为喜欢使用开放搜索，当搜索”kfss”时，doc1默认会排在doc2的前面。模糊搜索在用于查询意图不明确时能够很好满足用户的需求，但在使用过程中需要注意如下限制：

- 模糊搜索中按照空格分隔片段，认为按照空格分开的片段在语义上是等价的，比如对电影的演员进行

拼音搜索，多个演员之间是等价的，需要用空格分开。只有检索的内容处于同一个片段时，查询词两边才可以使用双引号，否则不建议使用。比如doc的short_text字段内容为‘刘德华 刘若英’，查询”ldh”或者”lry”可以把doc召回，而查询”ldh lry”或”liudehua liuruoying”是无法把doc召回的；
- 查询时只有英文、数字和拼音支持前缀和后缀搜索，中文不支持；
- short_text字段中的标点符号会被过滤掉；
- short_text字段过滤掉标点符号后，**长度限制为100个字节，超过的内容会被丢掉**；
- short_text字段可以创建下拉提示；
- 由short_text字段创建的索引不能够使用查询分析。
- 英文和数字及拼音不支持飘红

Schema

OpenSearch支持**多种数据类型及分词方式**，可以满足绝大多数场景下的需求。目前接触到的不满足的有：

- 目前支持的类型：INT、FLOAT、DOUBLE、LITERAL及其各自的ARRAY类型、TEXT、SHORT_TEXT；
- DynamicField：暂不支持，OpenSearch支持修改应用结构，可以先通过动态修改的方式来绕过；
- CopyField：暂不支持，可以离线预先合并。
- 字段个数限制：256。超过限制的业务可以考虑将非区间段查询（精准匹配）的若干字段利用ARRAY类型合并成一个字段，来减少总字段个数的方式绕过。
- patternTokenizer：目前OpenSearch支持自定义分词，但是分隔符默认为\t，需要将原有分隔符转化为\t即可。
- location：转化为两个字段float或者double字段，分别用来存储经纬度值。
- bool：转化为INT型，值为0/1。
- date：转化为INT型，数据库源会自动转成毫秒时间戳，API推送的需要手动转化；
- payload analyzer：暂不支持。
- bitwise分词：暂不支持。
- 庖丁分词：使用OpenSearch中文基础分词。

搜索语法

OpenSearch目前支持查询、过滤、统计、聚合、排序等功能，**详细功能说明。**

- q：必选参数，相当于OpenSearch中query查询，具体转化规则如下：

q 转化规则
‘:’暂不支持
range索引，用filter的区间段来转化
+A ==> A
-A ==> 不支持

A AND B ==> A AND B
A AND -B ==> A ANDNOT B
A OR B ==> A OR B
A OR +B ==> A RANK B
A AND B OR C ==> A AND B RANK C, e.g：红富士 AND 苹果 OR 山东
A OR B AND C ==> B AND C RANK A, e.g:红富士 OR 苹果 AND 山东
A AND B OR +C ==> A AND B AND C, e.g:红富士 AND 苹果 OR +山东
A OR +B AND C ==> B AND C RANK A, e.g:红富士 OR +苹果 AND 山东
+A OR B AND C ==> A AND B AND C, e.g:+红富士 OR 苹果 AND 山东
A AND B OR -C ==> (A AND B) ANDNOT C, e.g：红富士 AND 苹果 OR -山东
A AND -B OR C ==> A ANDNOT B RANK C, e.g：苹果 AND -红富士 OR 山东
-A AND B OR C ==> B ANDNOT A RANK C, e.g：-红富士 AND 苹果 OR 山东
A OR B AND -C ==> B ANDNOT C RANK A, e.g:红富士 OR 苹果 AND -山东
A OR -B AND C ==> C ANDNOT B RANK A, e.g:红富士 OR -山东 AND 苹果
-A OR B AND C ==> (B AND C) ANDNOT A, e.g:-红富士 OR 山东 AND 苹果
A OR B OR -C == A OR -C OR B == -C OR A OR B ==> (A OR B) ANDNOT C
A AND B OR C AND D ==> A AND B AND C AND D

- fq：用来过滤，只影响召回不影响算分。非模糊查询使用filter，模糊查询走query，排序的时候不要考虑该字段即可；
- fl：使用OpenSearch fetch_fields参数来定义返回值；
- hl：在控制台上配置结果摘要和飘红；
- start，rows： config 子句中的start和hit；
- wt：config子句中的format；
- df：查询的默认字段；
- sort：filed desc => -filed；field asc=> +field；score=>sort=RANK；
- facet：字段必须配置索引属性。

统计转化规则
facet.field => OpenSearch aggregate子句中的group_key参数
facet.limit => OpenSearch aggregate子句中的max_group，默认为1000
facet.mincount => 暂不支持，需要全部结果拿回去自行处理
facet.offset => 暂不支持，需要全部结果拿回去自行翻页
facet.sort => 暂不支持，需要全部结果拿回去自行排序
facet=true&facet.field=price&facet.limit=200 ==> aggregate=group_key:price,agg_fun:count(),max_group:200

- group：暂不支持。某些简单的场景可以考虑OpenSearch中的distinct子句，并结合sort来做组内排序。
- stats：部分功能对应OpenSearch中的aggregate子句，但是agg_func仅支持min, max, count, avg，暂不支持missing、sumOfSquares、mean、stddev、distinctValue、countDistinct。

搜索功能

- 深度翻页：目前OpenSearch提供两个查询接口，一个是search，一个是scroll。search是常规的查询场景，最多支持5000个结果返回，可以翻页，每页最大500个；scroll为数据导出场景，可以支持千万级别数据导出，但不支持排序，可以将结果拿回去做二次分析。
- 统计结果准确性：为了保证更优的检索性能，目前OpenSearch在很多情况下会做抽样和预估，这样会导致统计结果不是很精准。
- 搜索结果total值：为了保证搜索性能，数据量很大的情况下（跟总数据量无关，主要是查询召回量超过百万以上），仍然会做预估。
- 多OR查询：目前query长度限制编码后1K，如果OR查询较多会导致报错无结果，建议增加个数限制，或者并发多次查询再自行做结果merge。

性能篇

- 使用API/SDK推送数据有次数及大小限制，具体值请参考系统限制项，推荐将文档批量打包发送。
    - 数据上传后请务必检查返回值，并对相关错误码进行重试（尤其是3007错误），否则会出现数据丢失情况。同时，数据处理是异步的，系统返回“OK”后只表示系统接收数据成功，数据处理过程的错误会在控制台错误信息中展示，请注意及时检查。
    - CMD为“add”表示新增文档，会做全字段数据更新（更新中没有出现字段会默认为空）。如果该主键对应文档已经存在，则执行先“delete”再“add”的操作；
    - CMD为“update”表示更新文档，会做部分字段数据更新（更新中没有出现字段会仍为原来的值），针对已存在文档更新，也会先执行“delete”再“add”操作。如果未存在该主键值文档，则转成“add”操作。
    - CMD为“delete”表示删除文档，如果该主键对应文档已经不存在，则认为删除成功。
    - Post的数据大小有限制，如果您上传的文档过大（2M以上），服务器将拒绝接收任何参数，同时返回异常。
    - 当api推送报 connection timed out 或 1000:system error 或 push rate exceeds app quota 错误时，请重试直到成功为止。(注：push rate exceed app quota的报错不止是推送频率过高，更大概率是由于超过了每秒2M文档的限制而报错)。
    - 数据推送有次数限制，当api推送时，报错request too frequently建议批量打包发送，效率更高；
    - POST的url及body部分最好都要做url_encode，否则会出现解析及签名问题。
    - 数据源或者api推送增量时请注意，主键值重复的doc会被覆盖。

这里主要介绍在实际查询过程中可能遇到的各种情况，及可以优化的方法。当您发现自己的搜索效果不满意或者不知道该如何实现，请联系我们。

搜索查询的效果主要跟query关键词中命中的文档数有关，命中的文档数越多，系统要进行的计算就越多，那么耗时就会越高。所以优化的一个重要手段就是尽量降低query召回的文档数。

1. 查询需要带上索引名（应用结构中的“索引到”字段），否则将默认取default索引，如果没有default，则直接报错无结果。

```
query='mp3'相当于query=default:'mp3'
```

2. 查询关键词必须带上单引号，否则很可能会报错无结果。

```
Error:  query=default:mp3
Right: query=default:'mp3'
```

3. 如果查询词中包含’ ，则需要转义或者去掉；2，查询词的最后一个字符不能是’\’ ，否则会被当成转义符从而查询报错，如果要搜索’\’ ，需要对’\’进行转义。

```
query=default:'北京大学'，召回同时包含“北京”和“大学”的文档；
query=default:'abc's efg'会解析失败无结果，需要修改为'abc\'s efg'；
query=default:'abc\'，会查询报错无结果，\对'进行了转义，到时引擎解析失败；
```

4. 对于只用来做过滤筛选的需求，建议尽量将过滤字段建索引，通过query子句来查询，可以提高性能。

```
query=user_id:'123'&&filter=type_id=1，改写为query=user_id:'123' AND type_id:'1'。
```

5. 如果某些query召回量非常大，那么检索的效果低很多，这时候放在filter中可能效果更好。

```
query=status:'1' AND user_id:'123' //符合user_id=123的有100条记录，符合status=1的有5kw文档，
改写为
query=user_id:'123'&&filter=status=1 //这样召回量会小很多，通过filter来过滤会更快。
```

6. 在一些情况下，会查询近一个月的数据，目前系统不支持query的range查询，如果数据量较大，性能比较差，可以考虑增加一个月份字段，适当降低query的召回文档量，可以有效提高查询效率。

```
query=user_id:'123'&&filter=time>"2016-09-16" //符合user_id=123的有5千万数据，但是根据时间过滤完只有1千。query召回量太大，很容易超时
可以改为：
query=user_id:'123' AND (month:'201610' OR month:'201609')&&filter=time>"2016-09-16" //这样符合user_id:'123' AND (month:'201610' OR month:'201609')的只有2000，再做具体时间过滤效率会高很多。
```

7. 查询到具体的文档后，引擎会去获取实际要返回的结果数据，如果结果数据较大，那么消耗的时间也会越大。这时，可以从哪个方面入手：1，降低hit数，一般翻页结果为20个；2，修改默认展示字段或者fetch_fields，仅返回搜索结果展示中需要的字段即可。

工具篇

WordPress

基于RDS+OpenSearch实现WordPress站内搜索，有问题请论坛跟帖提问。

Discuz

基于RDS+OpenSearch实现WordPress站内搜索，有问题请论坛跟帖提问。

DedeCMS

基于RDS+OpenSearch实现WordPress站内搜索，有问题请论坛跟帖提问。

上线发布篇

在opensearch控制台中创建完应用后及后续维护应用时，需考虑到应用在后续运行中可能会因某些无法避免的因素（例如：数据错误，或其它原因）而导致应用行为不符合预期，所以需提前考虑到应对措施。

【高级版应用】

- 创建互备应用

应用出现故障后通常需进行索引重建修复数据，但该过程需要一些时间，因此建议再创建一个结构完全相同的应用，作为线上应用故障后的备份，即线上应用故障后，临时手动切到正常的备份应用中，避免依赖该应用的系统也出现长时间故障或服务不可用。

- 索引重建放在搜索低峰期

高级版应用索引重建数据是在原版本上更新，期间存在新老数据共存现象，任务完成后为最新数据，为避免对线上搜索服务产生较大影响，正常索引重建操作建议在搜索低峰期进行。

【标准版应用】

- 数据不兼容情况下人工切换新版本

相对于高级版，标准版应用支持多版本，通常情况下可配置”新版本全量索引构建完成后，服务自动切换到新版本”，但是，若存在新老版本数据不兼容的情况（如：有新增字段或字段值生成方法有变化，导致数据值无可比性），则需要在新建版本时配置”新版本全量索引构建完成后，人工切换到新版本”。该情况下用户需对新版本测试验证，符合业务预期后，再人工切换到线上提供服务。