开放搜索

最佳实践

最佳实践

功能篇

相关性实战

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 内置分析器 文档。

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。但可以通过查询分析解决。

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。

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：基础排序（粗排）和业务排序（精排），具体用法请点击此处参考文档。

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。

基础排序-（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。

业务排序-（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_relevance(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)))))	分类相关性、标题相关性、作者相关性 、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	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

	+ tag_match(ctr_query_value,doc_value,mul,sum,false,true) +..	

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。

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数组类型说明

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。插件文档

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。

FAQ

Q1: 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。

Q2: 有没有方法获得array类型的元素个数？
A: 系统提供了fieldlen函数，可以获取元素个数。

附近人搜索

OpenSearch支持类似附近人或地点的搜索。如果希望按照地点或附近人传入的坐标，那么可以使用本文介绍的方式，提高搜索效率，也同时提供排序功能。

解决方案

1. 配置GEO_POINT类型的应用结构字段以及地理位置索引，用于检索，召回结果。
2. GEO_POINT类型的字段设置数据源处理插件。
3. 搜索测试语法介绍，添加排序功能介绍。

配置步骤

1.创建应用结构配置

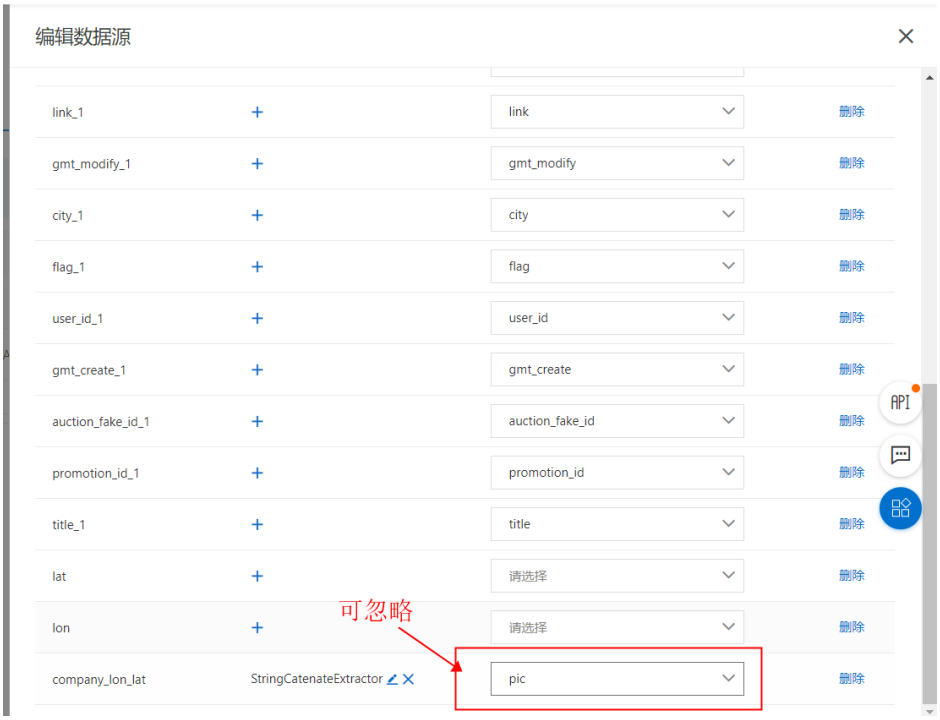
在OpenSearch**应用结构表中**增加lon和lat的DOUBLE类型地理位置坐标，再创建一个GEO_POINT类型的company_lon_lat字段（字段名称自定义）。**索引结构定义中**，为company_lng_lat设置分析器为地理位置分析器，并添加为属性字段。

完整创建应用流程请参考文档：快速搭建应用

2.数据源配置

字段映射时，将company_lon_lat这个应用结构字段配置StringCatenateExtractor插件，将现有的经度字段lon和纬度字段lat，联合起来。通过空格联合，生成到目标字段company_lng_lat。

注意：目标字段company_lng_lat的映射字段可忽略：



插件具体说明请参考文档：数据源及插件简介

3.搜索测试

例如：query=name:'Alibaba' AND company_lng_lat:'circle(116.5806 39.99624, 1000)'
说明：表示查询公司名为Alibaba，并且在坐标’ 116.5806 39.99624’ 附近1000米（1公里）以内的文档。
语法：query=spatial_index:'circle(LON LAT,Radius)'

- LON表示经度，LAT表示纬度，Radius为半径，单位米；半径10公里内性能最佳，超过10公里性能会大幅变差。
- 具体功能及语法可参考文档：range查询

4.新增精排表达式

例如：(distance(company_lng_lat,long_lat_in_query,distance)+1))*10000
功能：按距离远近将召回的文档进行排序。
语法：distance(location1, location2, outputname, defaultvalue)

- location1: GEO_POINT类型的字段名称
- location2: 用户查询串中kvpairs字句中设置的一个字段名，值为GEO_POINT字段要求的格式：LON LAT
- **outputname**：如果需要在结果中返回距离值，可以通过制定outputname值得到，如果不需要，可以不指定。
- **defaultvalue**: 用于指定当文档中location1的值非法时，distance返回的距离值，可以不指定，不指定时默认返回100000

说明：示例中long_lat_in_query需要在kvpairs中设置，例如：kvpairs=longitude_in_query:120.34256 30.56982

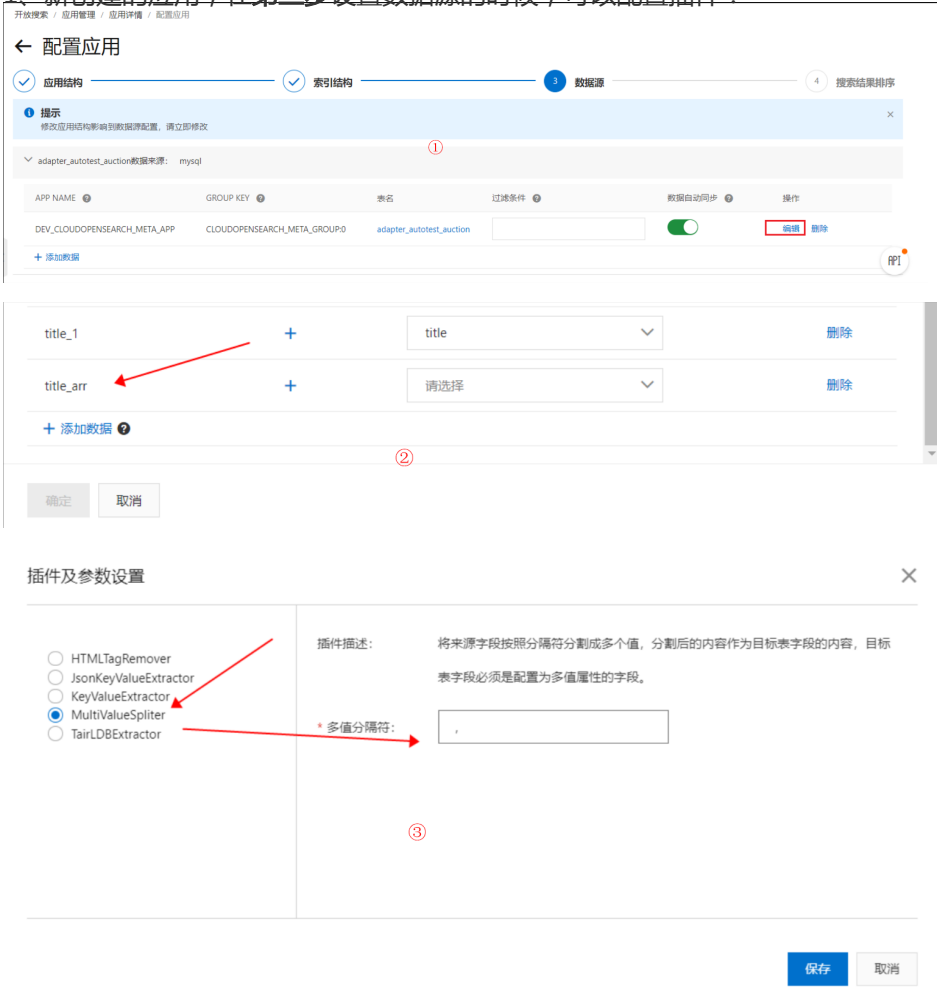
MultiValueSpliter插件的设置

注意事项

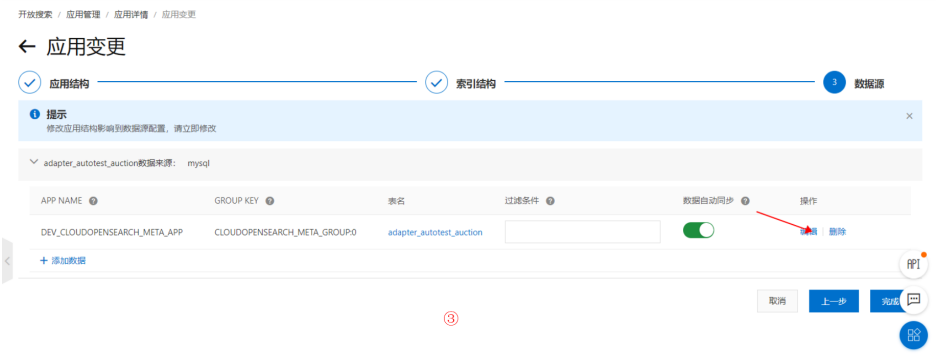
1. MultiValueSpliter插件是在数据源设置的地方设置；
2. MultiValueSpliter插件只支持literay_array字段；

MultiValueSpliter插件添加

1、新创建的应用，在**第三步**设置数据源的时候，可以配置插件：



2、已经创建好的应用（已线上应用为例），**应用详情页—->线下变更—->数据源配置**



查询分析——电商场景

在搜索中查询关键词的意图判断直接决定搜索到的结果是否可以满足需求。OpenSearch中查询语义理解

(Query Planner) 就是用来理解Query搜索意图的功能；通过对Query进行一系列智能分析，将Query进行改写后再在引擎中执行检索和排序。目前查询分析可选功能包括同义词拓展、停用词省略、拼写纠错、词权重分析、类目预测，除此之外在电商场景下还有实体识别的功能。下文将简单介绍查询分析各功能的基本介绍，以及给出电商场景中使用查询分析的具体样例。

停用词功能基本介绍

根据系统内置的停用词典过滤查询中无意义的词（一般是使用频度过高的但不影响查询结果的词，比如标点符号、语气助词等）。

拼写纠错功能基本介绍

用户输入的query并不总是正确的，错误的输入可能导致查询结果不符合预期或者是无结果，因此需要对用户的输入进行拼写检查。OpenSearch的查询分析中提供的拼写检查功能，对查询词中的错误进行纠正，给出正确的查询词。并根据纠错的可信度高低，决定当前查询是否用纠错后的词进行查询。

词权重功能基本介绍

该功能主要分析了查询中每一个词在文本中的重要程度，并将其量化成权重，权重较低的词可能不会参与召回。这样可以避免当用户输入的查询词中包含一些权重低的词时，仍然按用户输入的查询词限制召回，导致命中结果过少。

同义词功能基本介绍

在实际搜索场景中，会经常出现包含同义词的表达。例如，我们希望用户用户在搜索**苹果手机**的同时，包含**iPhone**的内容也能被检索并呈现。
在现实生活中，相同语义的表述词汇往往有很多，而用户在检索的时候很难在一条 query 中将它们全部体现，所以识别和提供同义词检索显然可以获得更高的召回率。
同义词功能主要是对查询词进行同义扩展，扩大召回和查询词同义的文档。

实体识别功能基本介绍

实体识别，全称命名实体识别（Named Entity Recognition，简称NER），指对查询词中的具有特定意义的语义实体进行识别。查询分析根据识别的结果，依据实体类型的权重对查询词进行改写，使得召回的文档符合查询的意图。Query改写主要根据实体的重要性，对query进行改写，召回时保留重要性高的实体词，对重要性低的部分不影响召回，只影响算法排序。

电商场景使用查询分析样例

以杨幂同款耐克修身连衣裙包邮.的查询词为例，不配置查询分析前的Query如下：

```
query=(default:'杨幂' AND default:'同款' AND default:'耐克' AND default:'修身' AND default:'连' AND default:'衣' AND default:'群' AND default:'包邮' AND default:'.')
```

配置停用词后实际系统查询词为：

```
query=(default:'杨幂' AND default:'同款' AND default:'耐克' AND default:'修身' AND default:'连' AND default:'衣' AND default:'群' AND default:'包邮')
//说明：此处停用词将查询词中的标点过滤掉了。
```

再添加拼写纠错后实际系统查询词为：

```
query=(default:'杨幂' AND default:'同款' AND default:'耐克' AND default:'修身' AND default:'连衣裙' AND default:'包邮')
//说明：此处拼写纠错将查询词中的错误输入“连衣群”纠正了。
```

再添加词权重后实际系统查询词为：

```
query1=(default:'杨幂' AND default:'同款' AND default:'耐克' AND default:'修身' AND default:'连衣裙' RANK default:'包邮')
query2=(default:'杨幂' RANK default:'修身' RANK default:'包邮' RANK default:'同款' RANK default:'耐克' RANK default:'连衣裙')
//说明：此处配置词权重后，权重较低的词“包邮”不参与召回。且当Query1无结果时，引擎会自动触发重查（re_search），按Query2召回结果，以避免无结果召回。
```

再添加同义词后实际系统查询词为：

```
query1=(default:'杨幂' AND default:'同款' AND ((default:'耐克') OR (default:'nike')) AND default:'修身' AND default:'连衣裙' RANK default:'包邮')
query2=(default:'杨幂' RANK ((default:'耐克') OR (default:'nike')) RANK default:'修身' RANK default:'包邮' RANK default:'同款' RANK default:'连衣裙')
//说明：此处配置同义词后，将“nike”添加为了“耐克”的同义词。且当Query1无结果时，引擎会自动触发重查（re_search），按Query2召回结果，以避免无结果召回。
```

再添加实体识别后实际系统查询词为：

```
query1=(default:'杨幂' AND ((default:'耐克') OR (default:'nike')) AND default:'修身' AND default:'连衣裙' RANK default:'包邮' RANK default:'同款')
query2=(((default:'耐克') OR (default:'nike')) AND default:'连衣裙' RANK default:'修身' RANK default:'包邮' RANK default:'同款' RANK default:'杨幂')
//说明：实体识别的结果：杨幂(人名)同款(后缀)耐克/nike(品牌)修身(款式元素)连衣裙(品类)包邮(营销服务)，
召回时保留重要性高的实体词，重要性低的部分不影响召回只影响排序。当Query1无结果时，
引擎会自动触发重查（re_search），按Query2召回结果，以避免无结果召回。
```

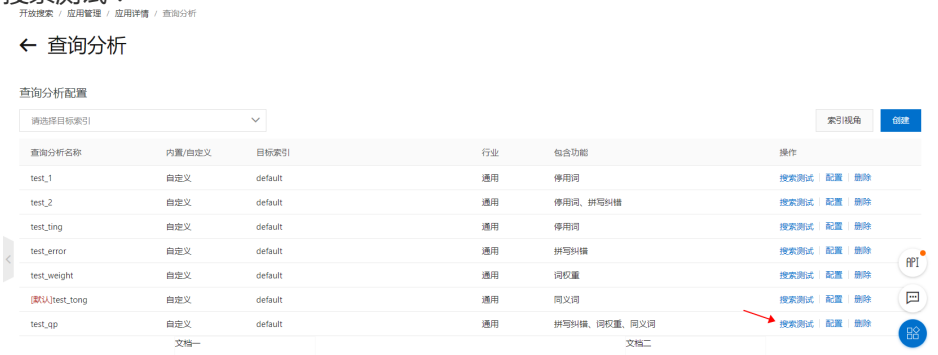
控制台配置流程

第一步：控制台——>**召回配置——>查询分析**，**创建**查询分析：

第二步：**配置**规则名称、索引范围、行业类型以及功能选择（可多选）。

第三步：3. 创建完毕后，可以进行**搜索效果测试**，调试无误后，**设置默认规则并对线上查询生效。**

搜索测试：



在索引视角下将其切换为默认：

solr语法转化

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。

- 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

数据推送

- 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。
  - 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 每秒提交的数据包总大小有限制，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，参考系统限制文档。
- 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。

- 使用RDS自动同步数据有TPS及大小限制，具体值请参考系统限制项：
  - RDS单库内所有表的更新会产生一份binlog数据。单库产生的binlog数据超过系统限制项，会造成数据堆积，同步延迟。建议拆库，将更新非常频繁的表放在不同的库里。如果不能拆库，建议主表走RDS自动同步数据，附表走API推送方式同步数据，来进行数据分流。

搜索

搜索优化

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“索引名”），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. 在一些情况下，会查询近一个月的数据，如果数据量较大，性能比较差，可以考虑使用range功能。

```
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 index_timestamp:(1473955200000,) //使用range查询效率会高很多。
```

6. 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。

解析查询结果

开放搜索产品，出于功能升级迭代需要，会不定期推出新功能，或对已有功能升级优化，实现用户的多样化功能或性能需求。

解析注意

基于产品需要功能升级迭代，查询结果中会根据需要，新增系统字段（比如 searchtime，viewtotal 之类的字段，就是系统字段）。

请确保在解析查询结果时，没有依赖查询结果中返回的字段个数，只是按需解析需要的字段。

行业篇

小程序场景

一、小程序搜索背景

前端：微信官方提供SearchBar插件，主要功能为搜索框的前端实现后端：实现简单的搜索分**小程序搜索**和**云开发两部分**，

- **小程序搜索包含以下三个接口，无商品、文本搜索相关能力：**

search.imageSearch：提供基于小程序的站内搜商品图片搜索能力search.siteSearch：提供针对页面的查询能力search.submitPages：提交小程序页面url及参数信息，让微信可以更及时的收录到小程序的页面信息

- **云开发**

开发者们可以使用云开发开发微信小程序、小游戏，无需搭建服务器，即可使用云端能力。云开发为开发者提供完整的原生云端支持和微信服务支持，弱化后端和运维概念，无需搭建服务器，使用平台提供的 API 进行核心业务开发，即可实现快速上线和迭代，同时这一能力，同开发者已经使用的云服务相互兼容，并不互斥。提供云函数、云数据库、存储、云调用能力。其中**搜索主要依赖云数据库实现。**搜索相关能力主要是可开发**基于正则的模糊匹配**，支持**基础聚合能力、GEO搜索能力、多字段模糊搜索最佳实践**，**但是搜索性能和搜索效果都有限，不支持分词、查询分析（QP）等能力。**

二、问题与需求

电商、内容行业对于查询意图理解有一定依赖的场景，使用小程序原生搜索服务功能较为基础，不足以支撑业务需求，用户搜索体验较差，则直接影响业务的转化。

三、电商搜索场景分析

搜索框输入效率低；

需求提升购买转化率；

关键词堆砌；

例如：搜索“帮宝适男童大码拉拉裤包邮” 分词后：帮宝适、男童、大码、拉拉裤、包邮； 一个query包含好几个搜索关键词**，这些关键词的重要性也是分：高、中、低

- **词序对语义的影响不大**；**

类目预测问题；

例如：当用户查询“苹果”时，可能查询的是水果，也可能是手机品牌。

四、内容搜索场景分析

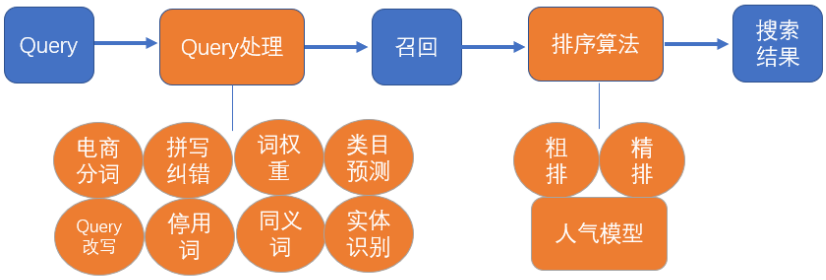
口语化描述的长尾词出现频率相对较高，关键词和内容多样性丰富、搜索用词规范程度参差不齐

- **聚合多个频道的内容**
- **要求搜索引擎具备较强的语义理解能力**

一部分场景垂直性专业性较强

例如：a. 搜索“有哪些适合男生看的科幻小说推荐”，Query涉及的关键词很多，聚合多个频道的内容，词序对语义影响不大b. 搜索“3岁以下宝宝咳痰怎么治疗”，专业性较强对搜索结果相关性要求更加精准

五、开放搜索解决方案



搭建需求	开放搜索
环境搭建	购买应用即可开始接入配置
数据接入	支持RDS、ODPS、POLARDB、API/SDK等多种数据接入方式
分词	内置阿里巴巴NLP分词，支持通用行业分词、电商行业分词、IT行业分词、中英文分词器等，查全率和查准率更高
查询语义理解	- 拼写纠错——纠正错误查询词； - 同义词——对中英文/近义词进行同义扩展； - 词权重——查询中每一个词在文本中的重要程度，并将其量化成权重； - 实体识别——电商行业中主要识别品牌、品类、款式、风格等实体类型，提高召回率和准确性，
排序	- 类目预测——根据查询词预测用户想要查询哪个类目的结果，结合排序表达式，使得更符合搜索意图的结果排序更靠前。 - 两轮相关性排序——第一轮为粗排，从命中的文档集合里海选出相关文档。第二轮为精排，对粗排的结果做更精细筛选 - 人气模型——量化出每个商品的静态质量及受欢迎的程度的值，不断训练统计形成人气分，构建更

	精细化的排序模型，精准命中搜索需求
下拉提示	在用户输入查询词的过程中，智能推荐候选query，帮助用户尽快找到想要的内容
热词底纹	- 热词——在搜索框下，根据搜索热度，获取最热的10条搜索词组/话题 - 底纹——在用户不输入搜索词时，直接显示在搜索框的优质query
算法功能	内置已成熟的多种高级算法功能
弹性扩展	简单操作，平滑扩缩容，即时生效
运维成本	0运维，免部署
开发部署成本	开箱即用，1位工程师0.5-1天即可完成基础服务搭建

六、小程序接入开放搜索仅需2步

6.1数据接入

- **数据源接入：**首先数据源接入是直接在阿里云购买rds(或者polardb等)，接入开放搜索。小程序和开放搜索的对接，仅仅需要查询的api接口或者sdk.
- **API/SDK接入：**小程序的接入跟app或者WEB的产品形态没有什么区别，只是微信小程序有自己的一套代码规范，但还是使用开放搜索的sdk/api来获取开放搜索的结果。

6.2搜索服务接入

- 通过在小程序JS里调用外部接口，获取搜索结果
- 普遍通过ES、Solr、第三方云服务等方式进行接入
- 示例如下：

```
wx.request({
url: '',
data: {SercherValue:that.data.inputValue,
SercherTypeNum:that.data.SercherTypeNum,
typeItem:that.data.typeItem },
header: {},
method: '',
dataType: '',
success: function(res) {},
fail: function(res) {},
complete: function(res) {},
})
```

电商行业

本文将介绍如何使用阿里云OpenSearch基于电商场景构建出一个简单的商品搜索原型，从而满足业务对商品搜索的需求。在构建类似电商平台建设中，有一块重要的业务要求是可通过关键字搜索的方式对商品信息中不同属性进行搜索，同时可对搜索出的商品列表进行分类的过滤，采用阿里云OpenSearch产品实现一个商品搜索的原型，能很好的满足项目的需求。

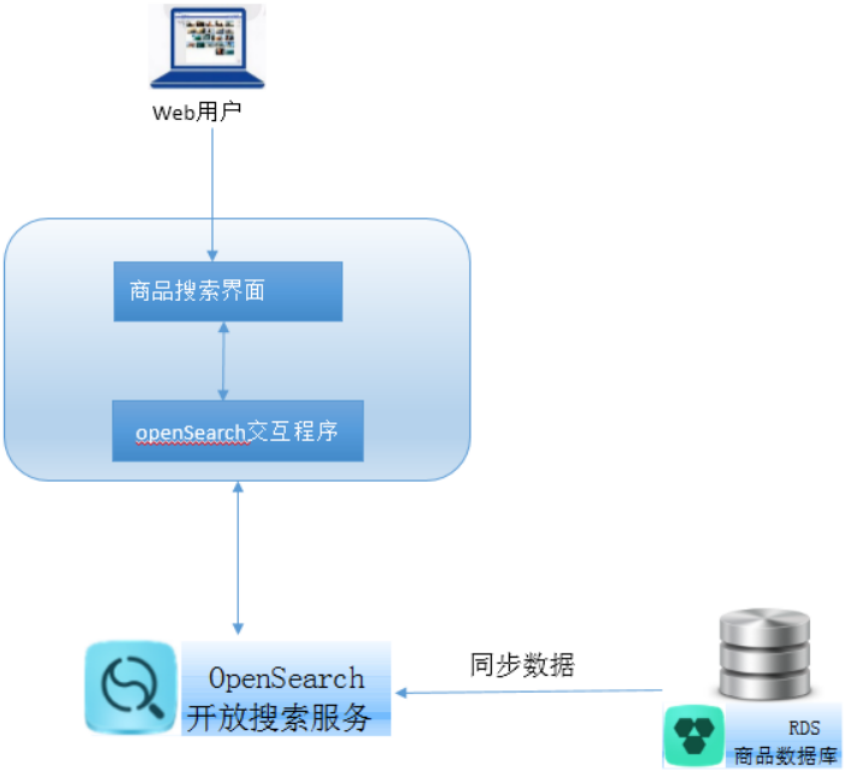
环境准备

第一次开通阿里云账号并登录控制台时，会提示先创建access key才能继续使用。

- 创建及使用应用依赖access key参数，主账号下access key参数不能为空。
- 在为主账号创建access key参数后，还可以再创建RAM子账号access key通过RAM子账号进行访问，RAM子账号赋予对应访问权限，请参考授权访问鉴权规则。

基本架构

整个原型的系统架构如下：



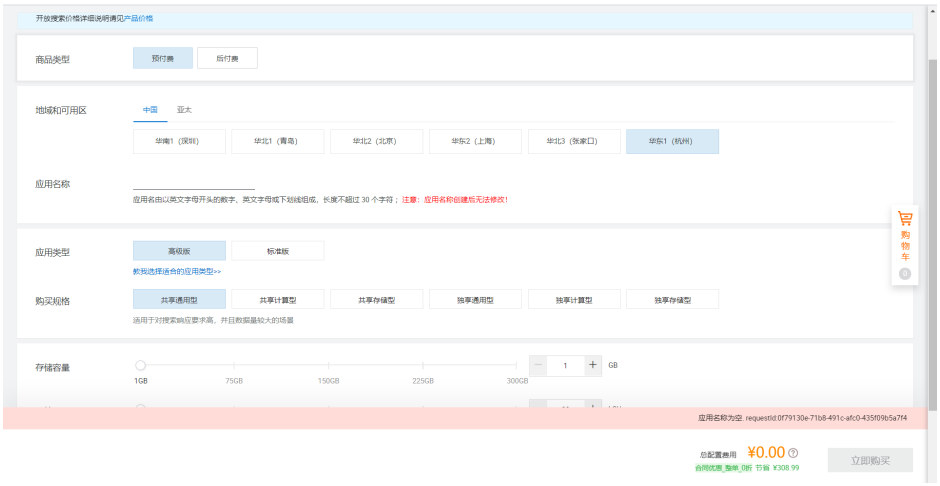
创建应用

1.登录阿里云OpenSearch控制台——->应用管理，单击右上角的**创建应用**：

2.选择应用类型：

本文基于电商场景下，涉及多表关系映射，所以选择**支持简单多表join逻辑的高级版应用**类型。关于应用类型的详细介绍请参考OpeanSearch应用类型说明。

3.填写应用信息：



选择商品类型：目前系统支持预付费和后付费，具体区别可参考：计费方式与价格。
选择地域和可用区域：目前opensearch支持的可用区域有

- 中国区域：华南1、华北1、华北2、华北3、华东1、华东2
- 亚太区域：新加坡

应用名：应用名由以英文字母开头的数字、英文字母或下划线组成，长度不超过 30 个字符 ；注意：应用名称创建后无法修改！
应用类型：高级版、标准版。
购买规格：目前opensearch支持的有共享通用、共享计算、共享存储、独享通用、独享计算、独享存储，详情可参考：计费方式与价格。
存储容量和计算资源：按实际而定，计算资源估算方法：LCU个数=QPS*单次查询消耗的LCU。其中单次查询消耗的LCU可以按照参考LCU估算器或者先购买一个入门型应用后，在搜索测试功能中进行测试查看单次查询消耗的LCU。

4.配置应用：

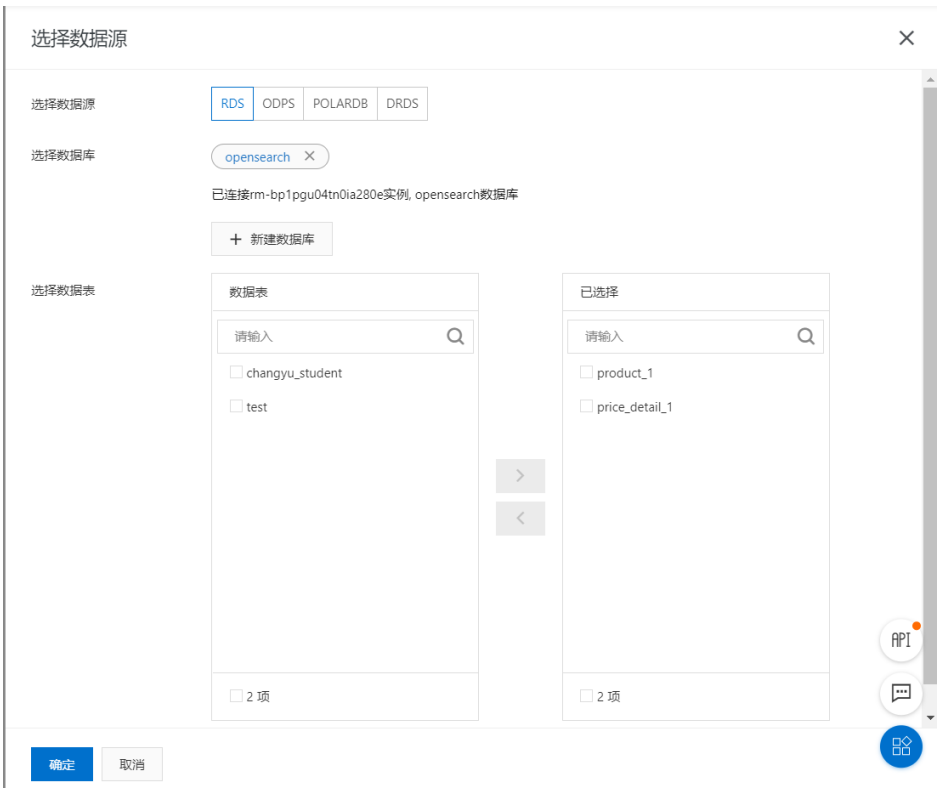
- **手动创建应用结构**：可以自定义应用结构进行应用创建。
- **通过模板创建应用结构**：系统默认提供了几种常用的模板样式，用户也可以将自己定义的应用结构创建成模板，可以通过已有模板快速创建出一个新的应用。
- **上传文档生成应用结构**：您可以上传已有的数据文件（仅支持JSON格式），系统会自动解析并创建出初始的应用结构（注意字段类型等需要重新定义）
- **通过数据源创建应用结构**：适用于通过RDS、MaxCompute(原ODPS)、POLARDB等数据源同步的场景，可以快速由源表结构创建出初始的应用结构，节省手动构造的工作量，降低出错概率。这里以RDS为例，其他数据源操作类似，具体详见**数据源配置**。

使用阿里云开放存储服务ODPS、RDS、POLARDB可以在OpenSearch控制台直接配置使用相应的数据源，数据将自动同步进入OpenSearch，简单、方便、可靠。本文将以RDS为例，选择通过数据源创建应用结构。

5.连接数据源

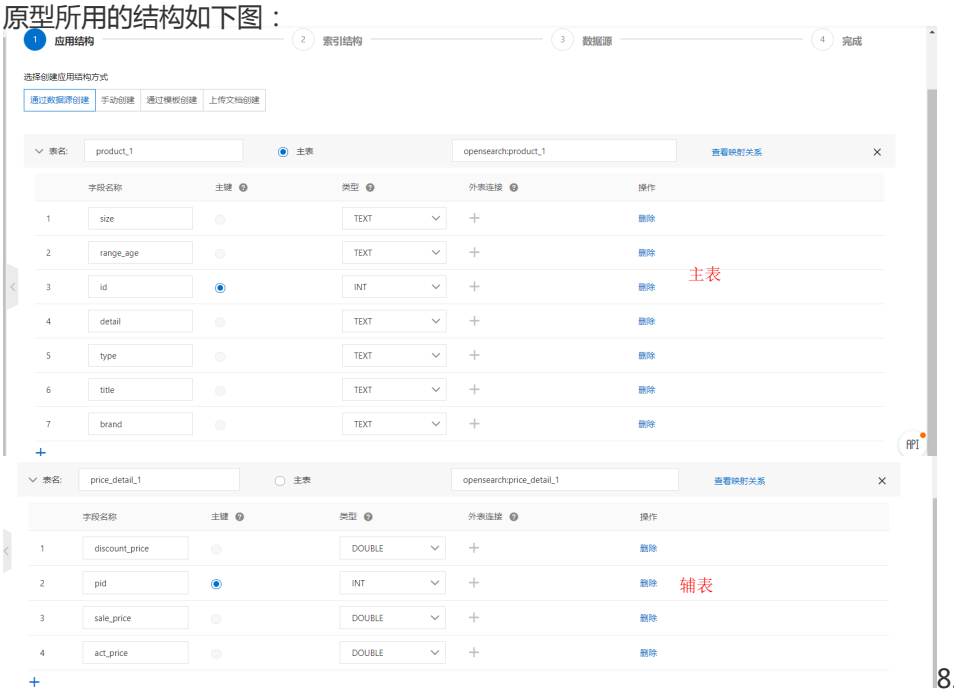
填写数据库配置，具体参见下图：

6.选择数据源：



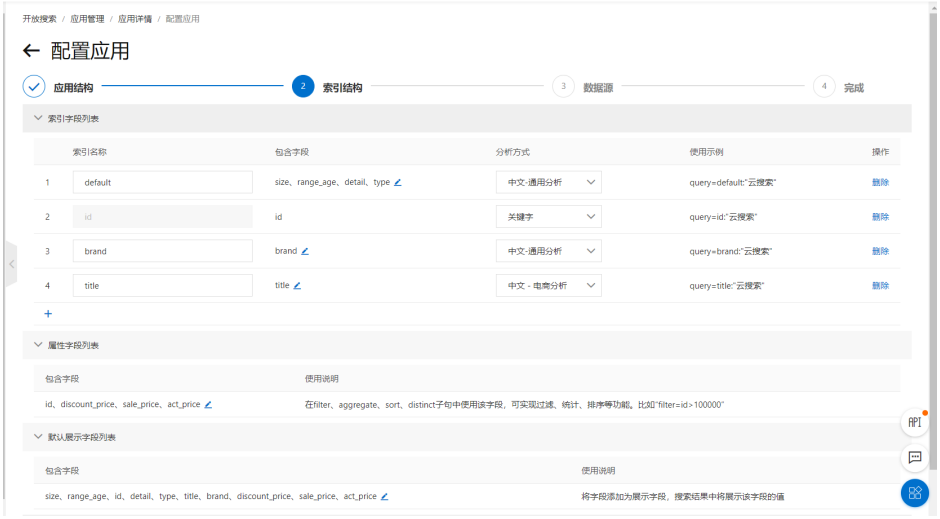
7.定义应用结构

我们采用商品表为主表，商品价格表为辅表的方式创建了应用结构即商品价格表主键pid关联商品表外键id。原型所用的结构如下图：



8.定义索引结构

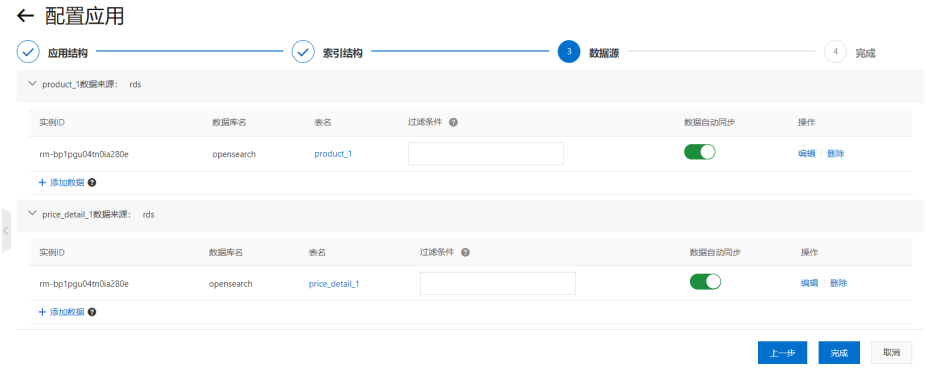
将商品表、商品价格表中需要搜索的相关字段添加到索引列表“default”中，这样就能通过query=default:“关键字”，对商品进行搜索操作了。具体配置见下图：



注意：分词方式直接影响搜索结果，请谨慎选择，关于分词方式的选择具体请参考文档。

9.配置数据源

在这里，您可以选择是否开启数据自动同步功能。 开启后，数据更新将自动同步至OpenSearch。



10.配置完成后，点击“完成”，此时在应用详情页中就可以看到，应用的状态处于“数据初始化中.”：

上传数据

上面我们是以RDS为例，索引构建中会默认开始导入全量数据，可以在应用详情页中看到具体进度。当然也可以调用OpenSearch API或者SDK来手动上传数据：

测试

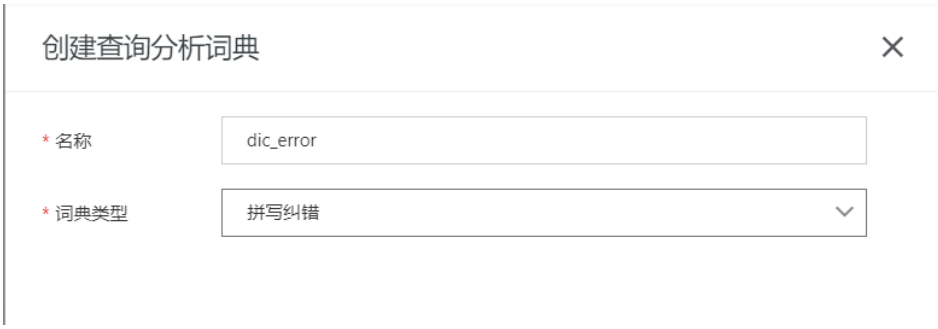
数据上传成功后就开始搜索体验了，OpenSearch控制台内置了OpenSearch搜索可以通过API/SDK或者页面搜索测试页面进行查询（详见API及SDK说明），本文以系统中搜索测试页面为例，搜索语法请参考API开发者手册搜索接口及搜索子句介绍。搜索结果如下图所示：

您还可以利用OpenSearch各种自定义的功能，来获得更优的搜索体验。例如解决目前用户长尾query召回少、搜索词填写错误无法召回、输入拼音无法召回等问题，具体请参考查询分析和相关性实战。

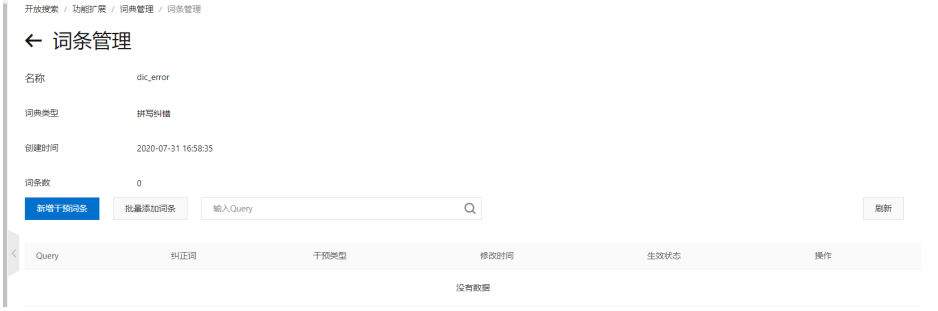
配置一个查询分析：
这里我们以拼写纠错为例配置一个查询分析：

第一步：创建查询分析干预词典：
1.1 依次单击控制台主页功能 扩展---->词典管理 进入查询分析干预词典页面：

1.2 点击右上角“创建”，词典类型为拼写纠错，输入词典名称点击“保存”：



1.3 在词典列表中刚刚创建的词典后单击**词条管理**进入干预词典页面：



1.4 单击**新增干预词条**配置您的干预词条：

1.5 单击保存，添加成功。您可以在干预词条列表中看到您创建的干预词条：

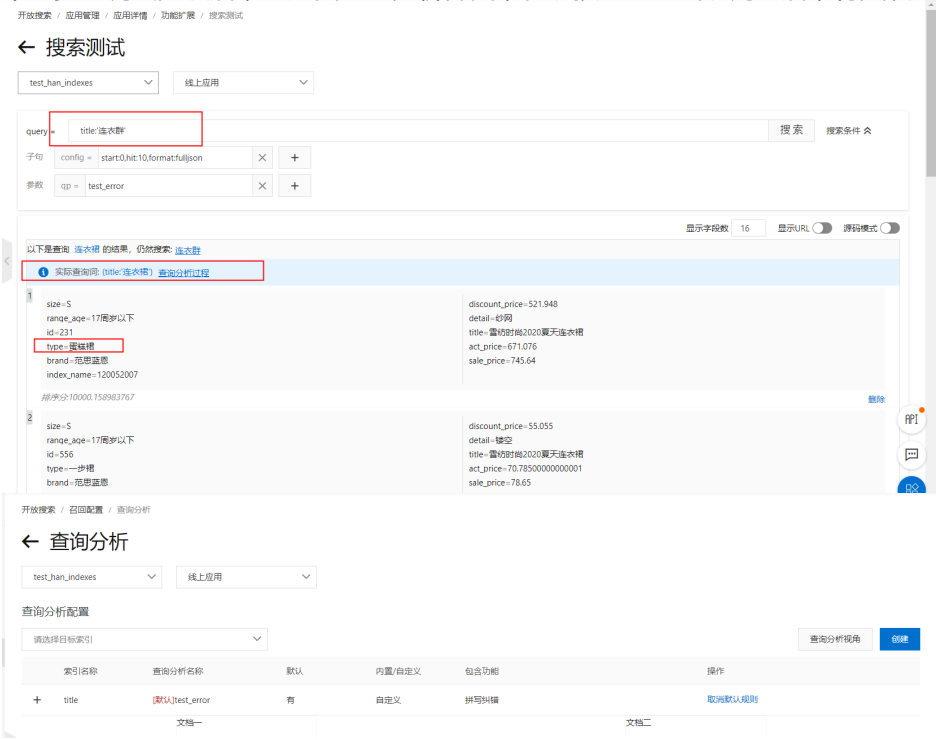
第二步：应用管理页面左侧导航栏中依次单击**召回配置→查询分析**进入查询分析页面：

第三步：单击右上角**创建**添加一个未上线规则，干预词典选择我们刚刚创建的dic_error：

- **停用词**：根据系统内置的停用词典过滤查询中无意义的词（一般是使用频度过高的但不影响查询结果的词，比如标点符号、语气助词等）。例如：查询词“奔跑吧!兄弟”，经过停用词处理后标点符号“!”不参与召回。
- **拼写纠错**：检查用户查询串中的拼写错误，并给出纠错建议。对于确定的拼写错误将直接改写原始查询串，然后进行检索；对于可能的拼写错误将仍然使用原始查询串进行检索。例如：查询词“阿里爸爸”，经过拼写纠错会改写为“阿里巴巴”，然后进行检索。
- **词权重**：分析查询中每个词的重要程度，并将其量化成权重，权重较低的词可能不会参与召回。例如：查询词“开放搜索好不好”，经过词权重处理，只要包含“开放搜索”的文档都可以召回。
- **同义词**：根据系统提供的通用同义词库和语义模型，对查询串进行同义词扩展，以便扩大召回。例如：查询原词为”KFC”，经过同义词处理后，包含”肯德基”或者”KFC”的文档都会被召回（配合词权重功能使用效果更佳）。
- **实体识别**：命名实体识别（Named Entity Recognition，简称NER）是系统对Query分词后将每个语义实体进行需求识别的功能。每个语义实体会被打上相应的类型标签，类型标签重要性低的语义实体在查询中可能会被省略；类型标签重要性高的语义实体会直接影响类目预测模型的训练。比如“耐克修身连衣裙”，实体识别的结果为“耐克/品牌/中”“修身/款式元素/低”“连衣裙/品类/高”。

第四步：创建之后，可在查询分析界面，点击“搜索测试”进行效果验证：

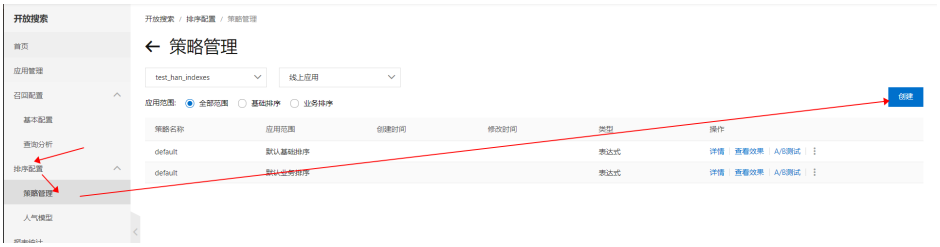
第五步：调试无误后，可返回查询分析界面，在切换到“索引视角”后，将其设置为默认查询分析：



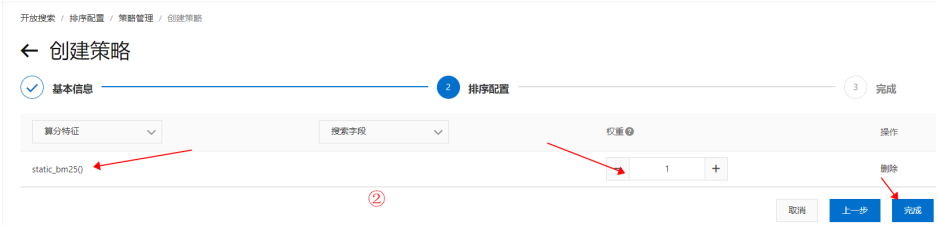
配置排序表达式：

排序表达式允许用户为应用自定义搜索结果排序方式，通过在查询请求中指定表达式来对结果排序，具体请参考搜索相关性配置。

第一步：依次单击应用主页排序配置→策略管理进入搜索结果排序管理页面：

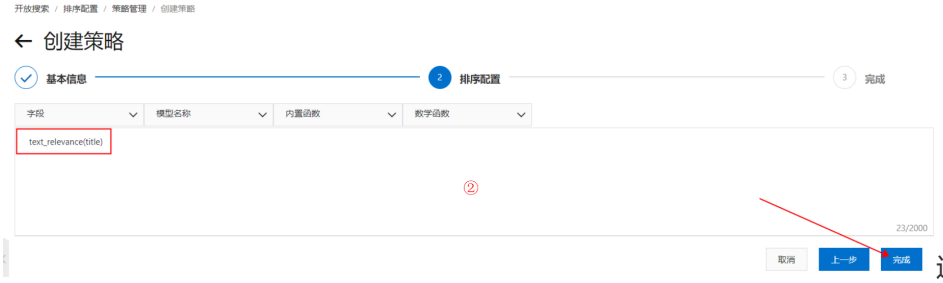


第二步：单击右上角的**创建**，添加新的基础排序（粗排）：



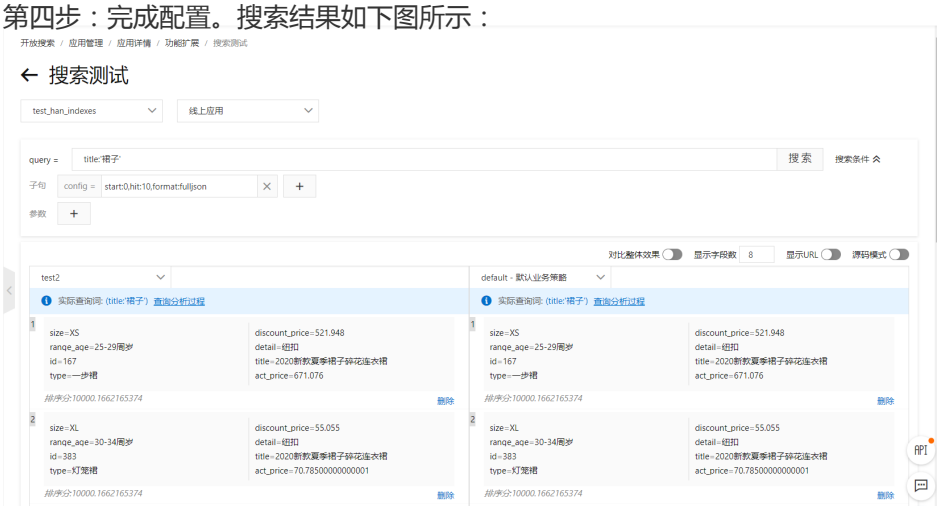
基础排序（粗排）对性能影响较大，尽量选用最有代表性的字段。这里我们以配置文本分、文档新旧程度的算分配置为例。

第三步：添加新的业务排序（精排）：



这里我们以配置商品名相关性相关性为例。

第四步：完成配置。搜索结果如下图所示：



这里搜索测试支持普通查询和设置策略排序后的查询结果对比。

配置下拉提示：

电商场景中，为了减少用户输入，帮助用户尽快找到想要的内容，我们可以使用开放搜索的下拉提示功能。关于下拉提示功能的配置请参考下拉提示。

配置类目预测：

您也可以使用类目预测功能预测用户想要查询那个类目的结果，具体步骤请参考类目预测。

其他常用配置：

- 电商场景中，如果某个商家的多个商品的文档分值都比较高，则都排在了搜索结果的前面。这样既不利于结果展示，也不利于用户体验。我们可以采用多样性聚合distinct子句来提升结果的多样性。具体用法请参考聚合子句-distinct。
- 业务上还支持按价格区间进行结果的查看，需要采用filter子句，更多用法请参考过滤子句-filter。我们以价格字段进行过滤为例，操作相关代码如下：

```
if(!lowPrice.equals("")){
queryElement.addFilter("price>=" + lowPrice);
}
if(!highPrice.equals("")){
queryElement.addFilter("price<=" + highPrice);
}
```

总结

至此一个简单的基于阿里云OpenSearch构建的电商场景下的搜索原型已经构建完成。OpenSearch提供了比较完善的搜索服务和API接口，能够基于OpenSearch快速实现业务对于搜索的需求，大大减少了开发工作，提高了搜索功能开发的实现效率，同时也减少了搭建复杂搜索引擎平台带来的系统部署和运维的工作量和成本。用户可以根据业务场景的不同选择OpeanSearch各种自定义配置和功能，从而获得更优的搜索体验。

工具篇

开源建站工具对接OpenSearch

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

上线发布需知

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】

- **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】

- 通常情况下可选择“需要，原线上应用将下线”，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），则建议在新建版本时配置“不需要，我将在线下应用详情页手动上线”。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。如图：