OpenSearch典型行业搜索解决方案

之 O2O

目录

OpenSearch典型行业搜索解决方案 1

之 O2O 1

1、概述 2

2、解决方案 2

2.1创建应用 5

2.1.1、配置数据表 6

2.1.2、配置搜索属性 7

2.2、数据上传 8

2.3、搜索效果配置 10

2.3.1 搜索相关性配置 10

2.3.2 搜索结果摘要 13

2.3.3 查询分析 13

2.3.4 下拉提示 14

2.4、搜索 15

2.4.1 初始列表页 15

2.4.2 排序功能 17

2.4.2 美食搜索 19

2.4.3 鲜花列表页 20

2.4.4 搜索小结 20

3、其他日常功能项 21

3.1 错误日志 21

3.2 索引重建 21

3.3 数据统计 21

# 1、概述

O2O即Online To Offline（在线离线/线上到线下），是指将线下的商务机会与互联网结合，让互联网成为线下交易的平台。O2O的概念已经落地，各大集团公司都陆续开始涉水，争夺势头愈演愈烈，各个行业遍地开花，如外卖、电影、wifi、机票、度假、酒店等，也正积极的影响着人民的生活，带来了极大便利性。这里的产品对搜索依赖比较重，大部分人的需求比较明确，比如机票起点终点、度假地点、饭店、酒店等，即使没有输入的初始列表页，因为需要根据距离等做一些排序，所以一般也会通过搜索的方式来获取，有效的做到千人千面的效果。

今天我们就以口碑来看如何利用OpenSearch来建立一个外卖检索系统。O2O具有很大的LBS共性，其他O2O行业的产品可以同类参考。

开始之前，先来个广告：）

口碑外卖-品质外卖，美食到家

口碑外卖是一款解决你“懒”,提供你品质服务的贴身应用。各类优惠活动，美食甜点，鲜花超市，你绝对会无情的为我点赞！ 当然，我的服务和体验绝对会让你喊嗨！

首先，你是一个人或者想尝鲜，再或者你太忙，还有“懒”，上口碑外卖就对了。特色小吃、美食外卖、甜点饮品、超市便利还有鲜花等等，只要你想要的，重点在吃，我都有！

其次，我想说，不管你在哪里，上班路上、游戏中、加班中、三亚海岛还是东北那嘎子，只要你想，我都在！上百万家的餐厅，上万家的大牌商家，满足你没问题！

超多品牌外卖如必胜客，肯德基，麦当劳，弄堂里，德克士，永和大王，均已入驻，各种美食，超市百货，鲜花蛋糕，中餐，西餐，小吃，快餐，火锅，烧烤，饮品甜点，让你足不出户就能叫外卖送餐上门。

# 2、解决方案

首先我们来看下口碑外卖首页，可以看到外卖中有很多分类：美食、甜点、生鲜、鲜花、医药等，非常丰富；排序与筛选条件也非常灵活，可以满足日常使用中的各种需求。目前仅提供了店名和菜品的独立搜索框，后续应该会有更多的入口开放出来。





鲜花频道列表页：



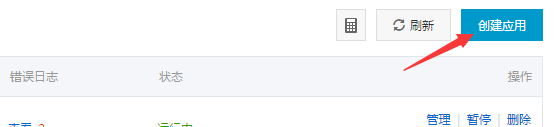
外卖及商品搜索：



建议开始前先粗览一遍[产品文档](http://help.aliyun.com/document_detail/opensearch/brief-manual/brief-summary.html?spm=5176.product8314983_opensearch.6.61.4AUuA4)，本文假设前提：您已经了解了OpenSearch的基本功能；您的场景适合使用OpenSearch；您接受OpenSearch的服务条款。关于OpenSearch的基本知识，不在本文讨论范围内，如有疑问，可以到[论坛](http://bbs.aliyun.com/thread/242.html?spm=5176.7189909.3.83.VQQMsk)或者[工单](https://workorder.console.aliyun.com/#/ticket/add?productId=1213)上提问。

要实现上图中的搜索功能只需要在OpenSearch中进行4步：

## 2.1创建应用

一个应用即为一个独立的搜索服务，简单理解就是一个搜索框。首先进入到[OpenSearch控制台](http://opensearch.console.aliyun.com/)，创建应用： 

填写应用名、选择好地域、备注：



点击下一步。

### 2.1.1、配置数据表

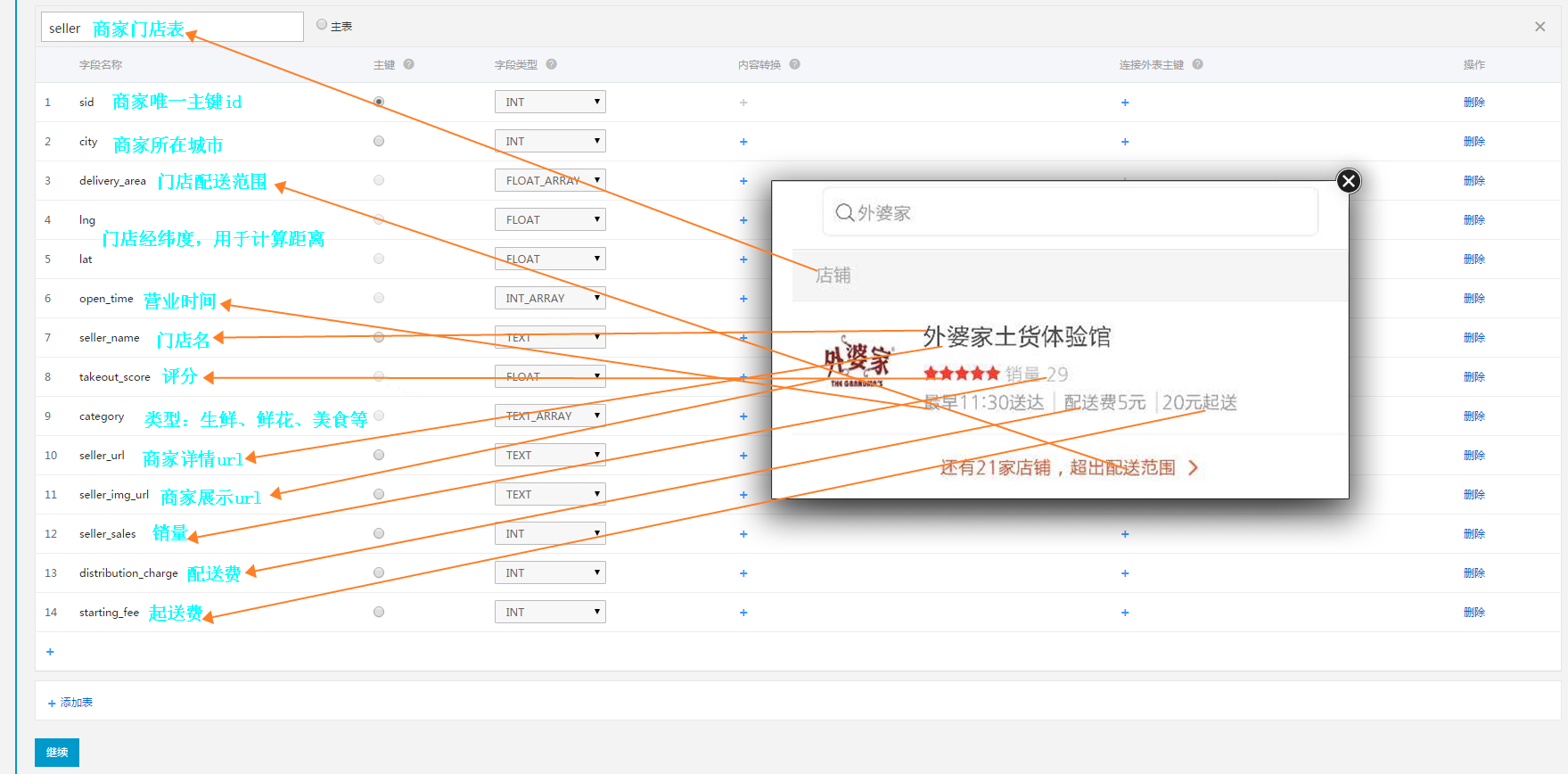
OpenSearch是结构化的搜索服务，首先我们需要定义表的结构，这里的字段一般是要用来查询、过滤、统计、展示的字段，其他无关字段无需进入OpenSearch中。以口碑外卖为例，我们定义应用结构如下：（本文主要是为了演示，只选取了必要字段，具体需要根据自己的场景来添加、修改或者删除）

这一步非常关键，很多同学并不知道如何配置，是否需要与数据库中字段一致，答案是不需要的，这里的字段均来自搜索需求。所以首先需要明确产品上需要做成的效果（如下图）、过滤（配送范围、营业时间等）、排序（距离、评分、销量、文本匹配度等因素）等需求。



从上图可以看出，页面展示的信息都必须在搜索中出现并且可展示，那么就出现了以下字段：





注意这里的关联关系，主表字段只能关联附表主键。

这里之所以使用了多表，是因为在实际业务中商家与菜品不会放在一张表中做存储，为了方便用户按原业务表推送数据，所以系统也支持了多表功能。当然，您也可以放在一张表，寻找最适合自己的方案即可。

如果配置错误也没有关系，系统提供了动态修改应用结构功能，您可以随时调整，不用担心。

### 2.1.2、配置搜索属性

多表只是为了方便推送数据，进入搜索引擎中的数据都是打平成大宽表的数据，所以搜索的时候不再有表的概念，这一步就来定义各个字段的搜索属性。这里是否要勾选需要根据实际搜索场景产品需求来定，另外细分类型非常重要，尤其是TEXT字段，[不同的子类型代表不同的分词效果](http://help.aliyun.com/document_detail/opensearch/operation-introduced/type-declare.html)，需要慎重选择。选择自己想要的，后续经过测试再不断的调整优化。

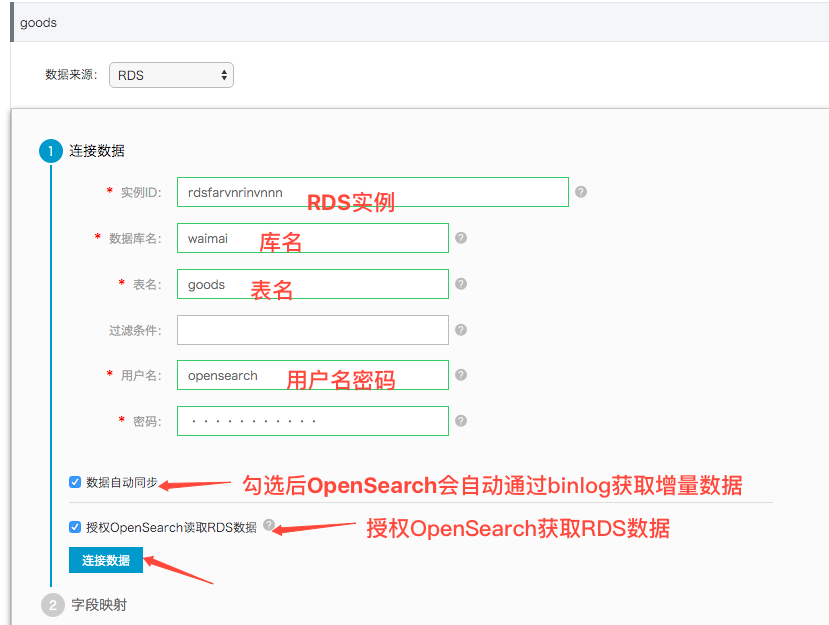


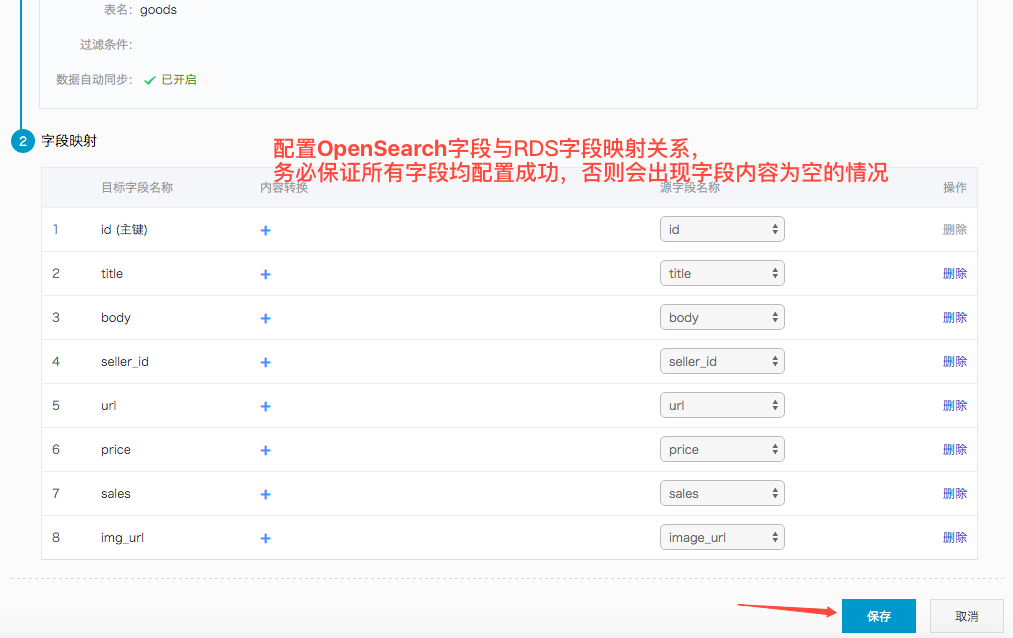
这里再重点说下可搜索与索引到，一个源段可以索引到多个索引字段上（如title索引到title、compound上），多个原字段也可以索引到同一个索引字段上（如title、body、seller\_name索引到compound上），具体如何配置同样是根据搜索需求来。产品上需要综合搜索，比如只要门店名、菜品名、菜品描述中有一项包含都希望搜索出来，那么可以将上述几个字段索引到同一个索引字段上，搜索的时候使用query=compound:’外婆家’，那么门店名、菜品名、菜品中有一项包含’外婆家’都会被召回。另外，产品上还希望提供按照菜品名搜索，那么可以单独将title索引到title上，那么query=title:’外婆家’只会召回菜品名中包含‘外婆家’的菜，门店中包含则不会被召回。

## 2.2、数据上传

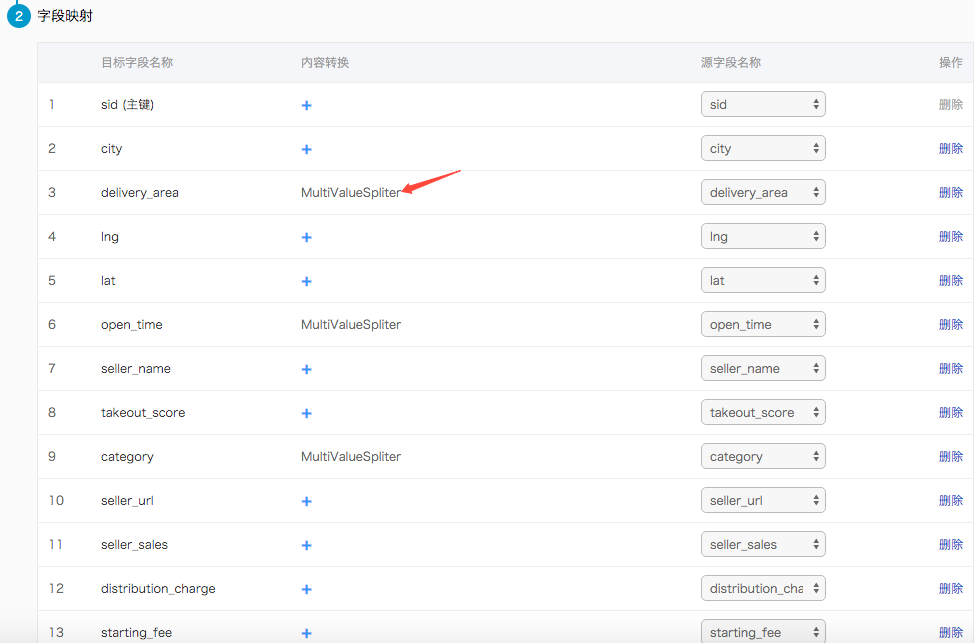
这里有两种选择，一种是后续通过API/SDK方式上传（具体参考产品文档的API/SDK参考手册），一种是通过数据源的方式导入，目前支持RDS\ODPS\OSS，这里以RDS源为例，其他类似（具体参考产品文档[数据源](http://help.aliyun.com/document_detail/opensearch/operation-introduced/basic/data-sync.html)配置）







同理，配置好seller表与RDS的映射关系（注意：array字段需要通过[MultiValueSpliter插件](https://help.aliyun.com/document_detail/opensearch/operation-introduced/basic/data-sync.html)来进行解析）



继续点击下一步，检查无误后即创建成功，按照提示进行应用激活以及索引重建操作（第一次RDS需要导入全量，后续走增量即可，无需再索引重建）。



## 2.3、搜索效果配置

经过上面的做法，数据已经上传完毕，可以搜索了。在搜索之前我们还需要进行下相关搜索效果的配置，已保证获取更优的展示效果。目前OpenSearch主要提供如下功能的支持，让我们来分别看一下：

### 2.3.1 搜索相关性配置

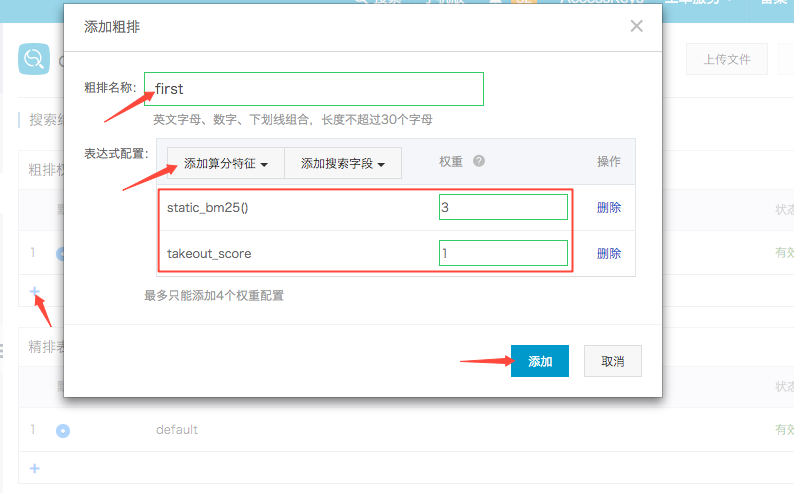
搜索引擎找到查询数据后，会对查询到的文档进行排序，并按照排序后的顺序返回给用户，用户可以采用翻页的方式来进行数据展示。目前OpenSearch支持[sort多维排序](http://help.aliyun.com/document_detail/opensearch/api-reference/query-clause/sort-clause.html)，在本文的案例中，我们需要使用到系统相关性表达式的功能。

目前系统提供的相关性开放了两轮的排序规则：[粗排与精排](http://help.aliyun.com/document_detail/opensearch/operation-introduced/senior/relevance.html)。粗排即是海选，将符合条件的文档进行粗排表达式计算得分，然后选取TOP N再进行精排表达式计算，最终按照精排得分排序后返回结果。从上面的处理逻辑来看，粗排因为涉及文档多，所以需要尽量简单（匹配的文档越多对于粗排表达式要求越高，否则会有好的文档进不了精排，无法展示的情况出现），将质量高的文档挑选出来即可，然后再根据精排来详细算分，做排序。



这里粗排表达式我们选取了两部分，二者之间的权重为2:1：

1. 首先，对于关键词查询，最重要的还是文本匹配度，所以这里选择了static\_bm25();
2. 其次，对于外卖来讲，店铺质量也非常重要，所以这里选择了takeout­\_score。

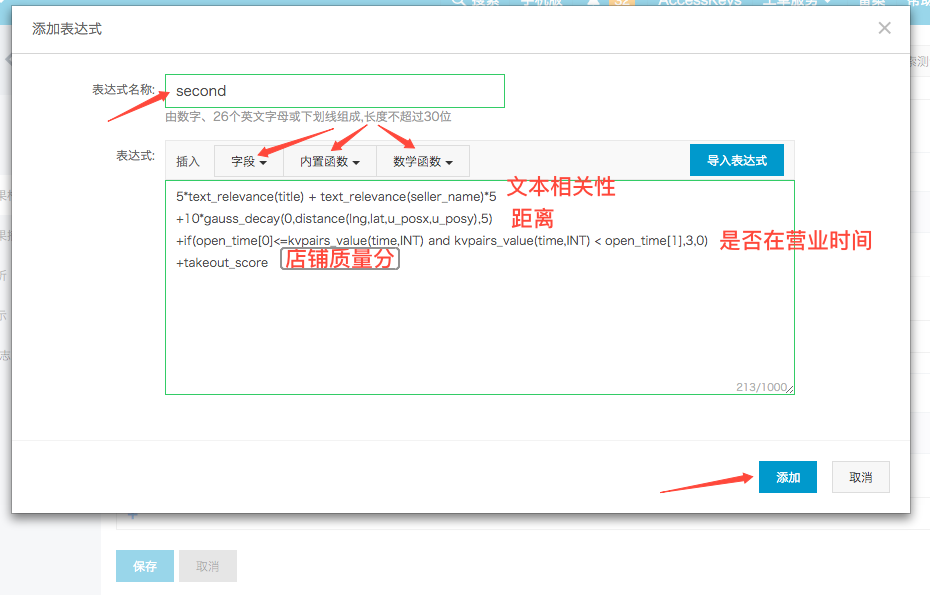


精排表达式如下，这里主要有如下几个因素考虑：

1，首先，对于一个文本匹配的场景，需要考虑文本匹配的程度，相关性越高分值越高；这里的text\_relevance()（包含三方面内容的考虑：bm25、proximity以及match\_doc度）。

2，其次，对于O2O场景来讲，距离是不可避免的考虑因素，距离越近理论上配送速度越快，用户体验越好，所以这里增加了distance()，并使用gauss函数对距离值进行衰减，使其值域统一到[0,1]上；

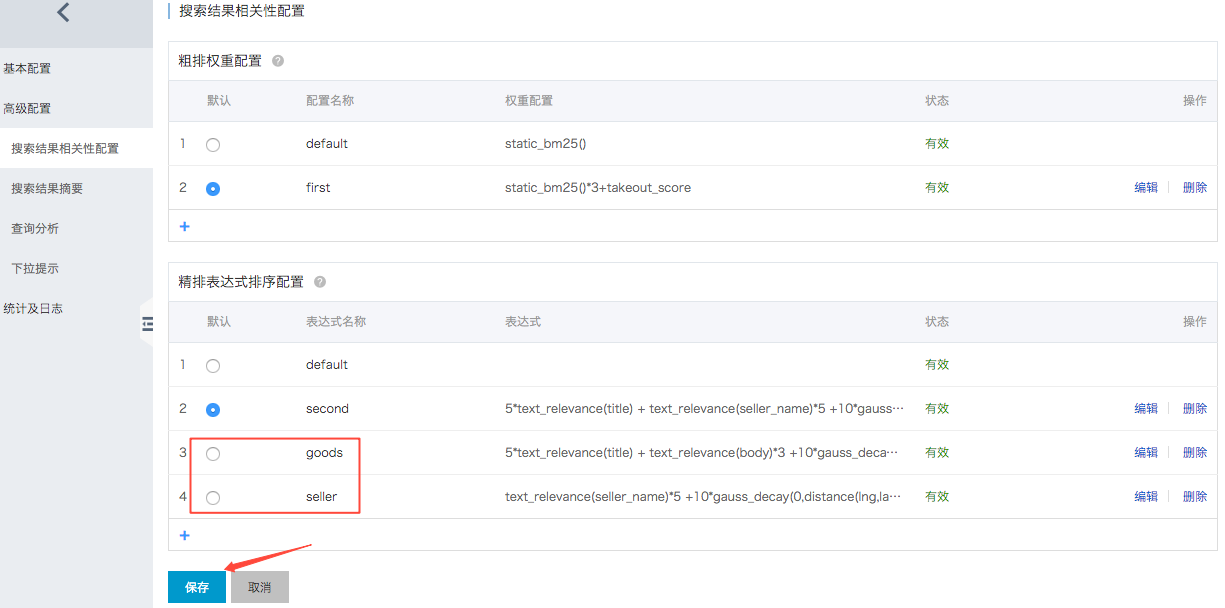
1. 对于外卖场景，还需要考虑门店是否营业，同样是影响配送速度的一个重要因素。所以，这里也增加了营业时间的判断。
2. 外卖店铺本身的热门程度也是一个非常重要的因素，所以这里也增加了takeout\_score的排序因子。这里的takeout\_score可以由很多因素来决定：品牌、受欢迎程度、用户评分、配送速度等等综合计算出来的一个店铺质量分。



确定主体排序因子后，根据各个排序因子之间值域以及重要程度进行合适的权重调整即可，这部分工作需要实际搜索效果中不断调试。目前相关性调试最常见的[ABTest](http://baike.baidu.com/link?url=EhDBX9cMeTqlm_x-f71U0MvaMqBbZR4JaFUUy6SaKOy4-rKTbkpPLXn8cv_l_eCJLAUOTJo-9CLecjzepaEpta)，寻找50%的TopQuery、20%长尾Query（即结果数非常少的查询）、30%常规Query（普通查询）进行效果人工排查，发现badcase后分析原因，然后调整排序表达式，以此不断优化，达到最佳效果。

设置成功后，别忘了点击保存按钮。一般来讲，一个新的排序表达式设置后可以先通过API接口显示指定first\_formula\_name以及formula\_name来进行效果测试，并根据测试结果不断调整，效果满意后，可以将其设置为默认即可。一旦设置为默认，则立即对线上查询起作用，无需发布变更，非常方便。

此外，对于较为复杂的搜索场景来讲，可以设置多个排序表达式以满足不同的查询场景。如下例中展示，搜索菜品的时候可以使用goods表达式，搜索门店的时候可以使用seller的表达式，综合搜索的场景使用second即可。



### 2.3.2 搜索结果摘要

摘要顾名思义就是文章的简介，一般来讲，文章内容都会比较长，在搜索结果中不可能全部展示，这时候就需要设置摘要信息，您可以配置改摘要的片段个数、片段长度以及片段连接符，片段选取跟查询词有关，通常选取关键词所在的一个或者多个句子。

同时，OpenSearch也提供了飘红功能，用来做查询词的高亮显示。



### 2.3.3 查询分析

查询分析主要是为了做查询优化的，他存在的目的就是为了深度解析查询词意图，并智能改写，比如一个查询地址：杭州市西湖区文一西路969号，能自动识别并改写成city:’杭州市’ AND district:’西湖区’ AND address:’文一西路969号’。当然这里只是举个例子，目前还未提供该功能，现在支持4部分：停用词（不参与搜索，如特殊符号和一些无意义的副词等）、同义词（如’肯德基’同义’KFC’等）、词权重（对一些不重要的词做RANK，扩大召回并提升搜索效果）、拼写纠错（如’百毒’纠错成’百度’等）。

您可以视具体的查询情况来配置上述功能，目前提供的还是系统自带词表，后续会开放用户自定义，一些较特殊的场景（如景点名、影视名等）可以自己配置适合的改写内容。



配置完毕后，可以通过API指定qp参数名字来测试（同上述排序表达式测试流程类似），观察改写效果，测试通过后，添加至上线，即可以对线上查询立即生效，非常方便。



### 2.3.4 下拉提示

下拉提示目前支持多个字段（最多3个）的语义单元抽取，目前抽取逻辑非常复杂，如果您发现有符合预期的情况，可以与我们反馈，我们会视情况来不断调整优化。

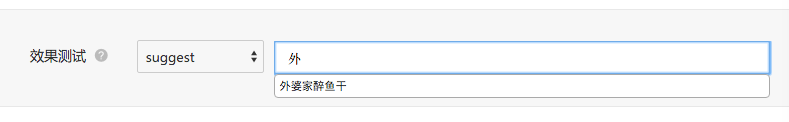
目前下拉提示对于抽取出来的语义单元进行拼音全拼与前缀等模糊搜索效果，可以很好的达到减少用户输入并规范用户输入提高搜索效果的效果。

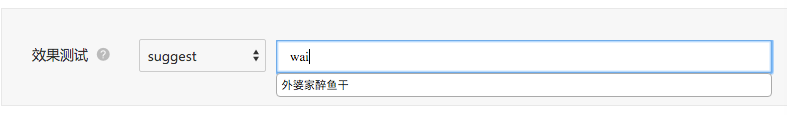


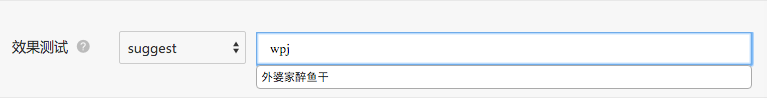
配置完毕后生效下拉提示，即可查看效果。



效果如下：







## 2.4、搜索

系统提供了丰富的[搜索语法](http://help.aliyun.com/document_detail/opensearch/api-reference/api-interface/search-related.html)：[查询](http://help.aliyun.com/document_detail/opensearch/api-reference/query-clause/query-clause.html)、[过滤](http://help.aliyun.com/document_detail/opensearch/api-reference/query-clause/filter-clause.html)、[统计](http://help.aliyun.com/document_detail/opensearch/api-reference/query-clause/aggregate-clause.html)、[排序](http://help.aliyun.com/document_detail/opensearch/api-reference/query-clause/sort-clause.html)、[打散](http://help.aliyun.com/document_detail/opensearch/api-reference/query-clause/distinct-clause.html)等。

接下来我们来看下如何调用OpenSearch API来实现口碑外卖的效果。

### 2.4.1 初始列表页



用户刚打开页面出现的页面，此时用户并没有进行搜索，所以需要制定规则筛选出一批商品展现给用户，这里筛选策略可以自己指定，比如：用户购买过的店优先；距离近的优先；好评度高的优先；近期优惠力度大的优先等等，以达到吸引用户点击的目的。

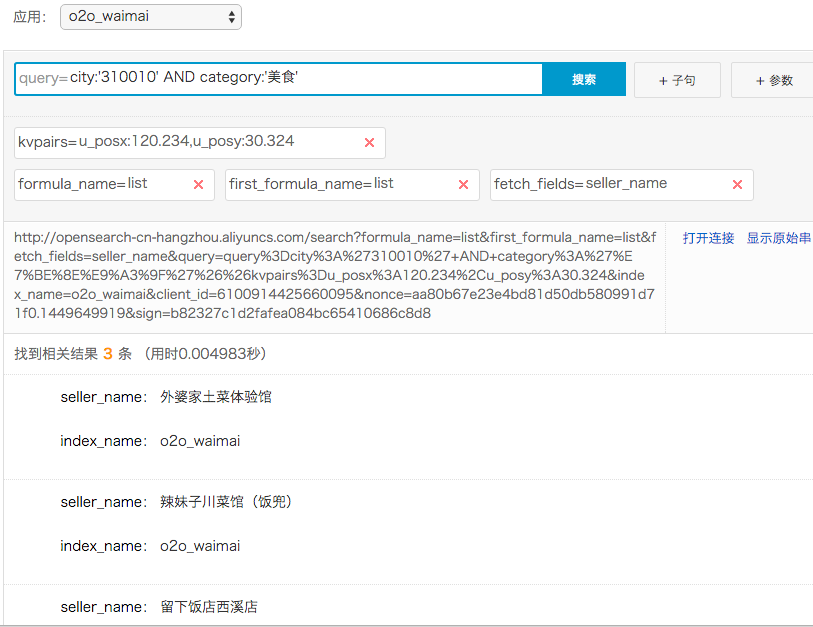
这里我们按照距离、好评等因素来综合来排序；美食的用户量最大，所以这里只展示‘美食外卖’的门店：

|  |
| --- |
| query=category:’美食’  //查询类型为’美食’的门店  sort=-RANK  //按照搜索相关性配置进行排序  kvpairs= u\_posx:120.234,u\_posy:30.324  //通过用户位置获取经纬度坐标，并通过kvpairs中的u\_posx、u\_posy参数传递给formula用来计算用户与商家的距离值  first\_formula:takeout\_score  //通过门店质量分进行门店海选  second\_formula:takeout\_score+10\*gauss\_decay(0,distance(lng,lat,u\_posx,u\_posy),5)  //对海选出来的门店取TOP N个进行精细排序，主要考虑门店质量分以及距离 |

另外，在实际外卖的场景下，用户肯定是在当前城市中的查找，我们可以考虑通过city来缩小查询范围提高效率，上面查询改写为：

|  |
| --- |
| query=city:’310010’ AND category:’美食’  //查询类型为’美食’且城市编号为310010的门店 |

可以在搜索测试页面进行下初步搜索效果查看：



观察没问题后，在程序中调用API接口获取上述结果渲染成最终的产品样式即可。

### 2.4.2 排序功能



在上面搜索的基础上进行排序修正：

按照销量排序：

|  |
| --- |
| sort=-seller\_sales |

按照最早送达时间排序：

|  |
| --- |
| sort=-RANK  first\_formula:takeout\_score  //按照门店质量分粗选  formula：if(open\_time[1]>time, open\_time[1]+prepare\_time,  //如果起始时间超过当天时间，则说明还未营业，则送达时间是开始营业时间+准备时间  if(open\_time[1] <=time and open\_time[2], prepare\_time,  //如果当天时间在营业范围内，则送达时间即为准备时间  10000))  //如果超过了营业结束时间，则无法送达，这里设置了一个MAX值10000  //open\_time[1]中是配送起始时间距离当天0点的分钟数，open\_time是配送结束时间距离当天0点的分钟数。如营业时间是[9点,21点]，那么open\_time[1]为9\*60=540，open\_time[2]为21\*60=1260。同理，time是当前时间距离当天0点的分钟数，比如当前是10点，则time为600。如果一天中多个营业时间段，则类似处理。prepare\_time是准备及配送时间，可以由商家根据实际情况提供。 |

按照距离远近排序：

|  |
| --- |
| sort=distance(lng,lat,”120.234”,”30.323”)  //120.234、30.323为用户坐标的经纬度 |

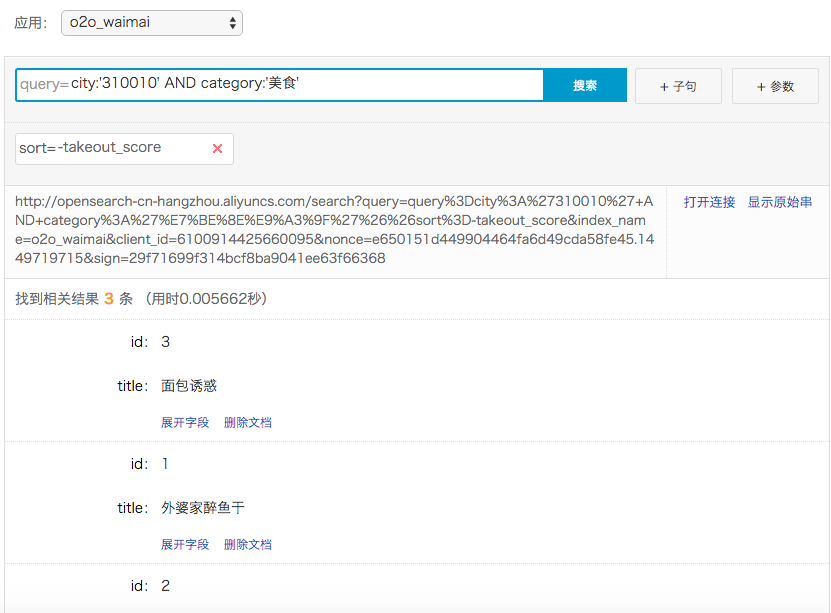
按照起送价由低到高排序：

|  |
| --- |
| sort=+starting\_fee |

按照评分由高到低排序：

|  |
| --- |
| sort=-takeout\_score |

搜索测试页面效果：



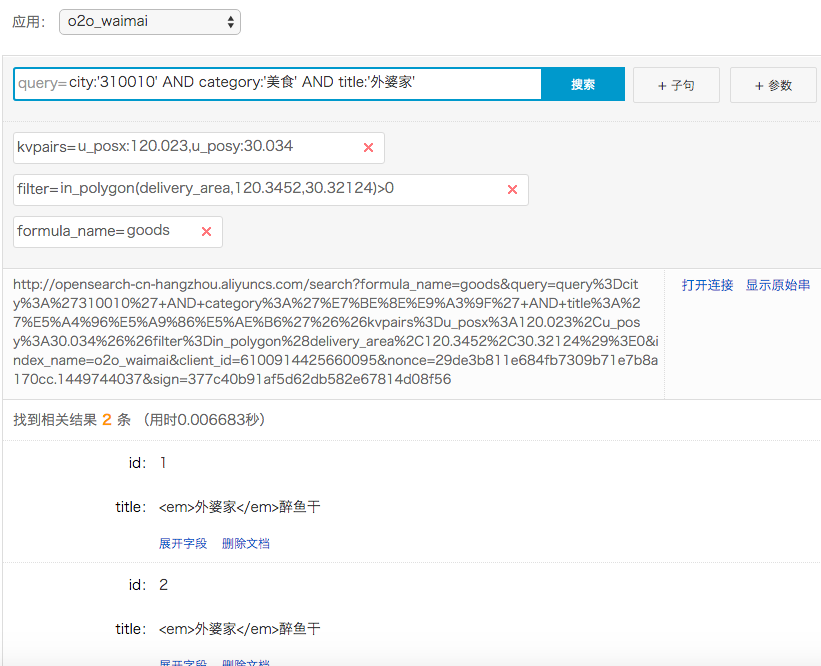
### 2.4.2 美食搜索



这里增加的搜索词，并且从上面效果图可以看出，结果展示的时候区分了店铺和商品两项，所以查询的时候需要分开做查询：

|  |
| --- |
| query=city:’310010’ AND category:’美食’ AND seller\_name:’外婆家’  //查询类型为’美食’且城市编号为310010的门店  query=city:’310010’ AND category:’美食’ AND title:’外婆家’  //查询类型为’美食’且城市编号为310010的菜品  filter=in\_polygon(delivery\_area,120.3452,30.32124)  //需要在配送范围之内的，其中（120.3452,30.32124）为用户坐标  kvpairs= u\_posx:120.234,u\_posy:30.324  //通过用户位置获取经纬度坐标，并通过kvpairs中的u\_posx、u\_posy参数传递给formula用来计算用户与商家的距离值  sort=-RANK  first\_formula: static\_bm25()\*3+takeout\_score  second\_formula: 5\*text\_relevance(title)  +10\*gauss\_decay(0,distance(lng,lat,u\_posx,u\_posy),5)  +if(open\_time[0]<=kvpairs\_value(time,INT) and kvpairs\_value(time,INT) < open\_time[1],3,0)  +takeout\_score  //设计初衷见排序表达式部分的介绍，这里就不额外赘述了。 |

搜索测试页面效果如下：



### 2.4.3 鲜花列表页

实现方式与美食类似，只不过修改下查询query即可，这里不再重复介绍。

|  |
| --- |
| query=city:'310010' AND category:'鲜花' |

### 2.4.4 搜索小结

调试过程中会遇到一些报错请，请根据错误码查找[文档找到错误原因](http://help.aliyun.com/document_detail/opensearch/api-reference/call-method/errormsg.html)进行修改即可。测试过程需要根据实际情况对排序表达式、应用结构不断进行调整，以便达到满意的效果。

另外，在使用中有很多[搜索性能优化项](https://help.aliyun.com/document_detail/opensearch/best-practice/performance/search.html)也可以参考。

测试完毕后，便可以调用[API语法](http://help.aliyun.com/document_detail/opensearch/api-reference/terminology.html)（OpenSearch提供了[php SDK](http://help.aliyun.com/document_detail/opensearch/sdk/php/preface.html)、[java SDK](http://help.aliyun.com/document_detail/opensearch/sdk/java/preface.html)）进行实际的搜索结果页开发了。这里就不做深入讨论，论坛上有篇《[OpenSearch phpsdk从到初级到高级的详细教程](http://bbs.aliyun.com/read/169969.html)》各位可以参考，有问题可以联系我们。

# 3、其他日常功能项

## 3.1 错误日志

数据处理过程是异步的，数据成功上传到OpenSearch后，会经过若干处理模块最后进入引擎做索引构建（具体查看[OpenSearch基础架构](http://help.aliyun.com/document_detail/opensearch/brief-manual/principles.html)），在该处理过程中可能会因为权限、类型、插件等各种原因导致处理失败，这部分失败会在控制台错误日志处有详细展示，您可以根据[错误码和错误提示](http://help.aliyun.com/document_detail/opensearch/api-reference/call-method/errormsg.html)及时修正数据，重新上传。



## 3.2 索引重建

用户数据上传后会先保存在离线HBASE中，然后经过相关处理后进行索引构建到引擎，供用户查询。如果有应用结构变化，那么需要将数据以新的结构定义重新进行索引构建。这个构建过程是流式的（如控制台索引重建上的进度条所示），新数据会逐渐替换老数据，所以在索引重建过程中会出现新老数据共存的现象，索引重建完毕后新数据完全生效。

在这个过程中，搜索服务不受影响，但是因为新老数据共存，所以需要保证查询的可用性

1，如果是新增字段，那么原字段的逻辑都是兼容的，建议在索引重建完毕后再访问新字段

2，如果是删除字段，建议在索引重建前先去掉对已删除字段的访问；

3，如果是修改操作（如类型），那么需要保证查询对于修改前后的都是通用的。否则，险起见，建议重新创建新应用，待新应用数据完成后，再将搜索切换到新应用上，删除老应用。

注意：RDS源用户只要第一次索引重建将全量数据导入即可，后续系统会自动通过binlog做增量数据更新，无需定时或者手动重建。

## 3.3 数据统计

应用的访问PV、文档数的运营数据指标都可以通过数据统计来展示，方便随时掌握应用变化情况。