Elasticsearch 提高查准率

检索类型选型

**1**，**match**检索

含义：精细化切词匹配，只要待匹配的语句中，有一个满足检索结果，就能匹配到。

场景：结果可能达不到实际开发预期。实际业务中但凡有精准度要求的都较少或几乎不使用。

举例：

1PUT doctor\_index/\_doc/4  
2{  
3  "content":"刘强东方才只是睡觉了,并没有违法"  
4}

我输入检索词

“小王睡觉”

，也能匹配到上面的content。

**2**，**match\_phrase:**短语匹配

含义：相比match，更强调多个分词结果和顺序，都要完整匹配才能检索到。  
场景：实战应用中会较多使用，结合slop调整顺序和精度。

slop例：

为了让查询 quick fox 能匹配一个包含 quick brown fox 的文档

GET /my\_index/my\_type/\_search

{

"query": {

"match\_phrase": {

"title": {

"query": "quick fox",

"slop": 1

}

}

}

}

slop 参数告诉 match\_phrase 查询词条相隔多远时仍然能将文档视为匹配 。 相隔多远的意思是为了让查询和文档匹配你需要移动词条多少次

Pos 1 Pos 2 Pos 3

-----------------------------------------------

Doc: quick brown fox

-----------------------------------------------

Query: quick fox

Slop 1: quick ↳ fox

通过设置足够大的 slop 值， 查询单词能按照任意顺序排列，而不必考虑实际文档中的单词排练顺序。

为了使查询 fox quick 匹配我们的文档， 我们需要 slop 的值为 3:

Pos 1 Pos 2 Pos 3

-----------------------------------------------

Doc: quick brown fox

-----------------------------------------------

Query: fox quick

Slop 1: fox|quick ↵

Slop 2: quick ↳ fox

Slop 3: quick ↳ fox

|  |  |
| --- | --- |
|  | 注意 fox 和 quick 在这步中占据同样的位置。 因此将 fox quick 转换顺序成 quick fox 需要两步， 或者值为 2 的 slop 。 |

**3** ，**query\_string**

含义：支持与（AND）、或（OR）、非（NOT）的匹配。

场景：有与或非多值匹配的场景，无需单独再开发，开箱即用。底层的关键词实际走的是match\_phrase，

不过多个参数（如：default\_operator，phrase\_slop等）可控制调整精度。  
举例：

1GET /\_search  
2{  
3    "query": {  
4        "query\_string" : {  
5            "default\_field" : "content",  
6            "query" : "刘强东 AND 无罪"  
7        }  
8    }  
9}

检索不到的原因分析

**1**，分词原因**/**词典未收录原因

例：以下查询无匹配

1PUT doctor\_index/\_doc/3  
 2{  
 3  "content":"佟大为老婆生了孩子"  
 4}  
 5POST doctor\_index/\_search  
 6{  
 7"profile":"true",   
 8  "query": {  
 9    "match\_phrase": {  
10      "content": "佟大"  
11    }  
12  }  
13}

GET /\_analyze #查看切词  
2{  
3  "text":"佟大为老婆生了孩子",  
4  "analyzer": "ik\_max\_word"  
5}

|  |  |  |  |
| --- | --- | --- | --- |
| token | start\_offset | end\_offset | position |
| 佟大为 | 0 | 3 | 0 |
| 大为 | 1 | 3 | 1 |
| 大 | 1 | 2 | 2 |
| 为 | 2 | 3 | 3 |
| 老婆 | 3 | 5 | 4 |
| 老 | 3 | 4 | 5 |
| 婆 | 4 | 5 | 6 |
| 生了 | 5 | 7 | 7 |
| 生 | 5 | 6 | 8 |
| 了 | 6 | 7 | 9 |
| 孩子 | 7 | 9 | 10 |

分析可知：佟大两个字组成的连词，没有作为词组分配的，所以匹配不到。

**2**，**postition**位置不一致。

假定我字典里面没有收录“刘强东”这个人名。  
举例：

1PUT doctor\_index/\_doc/4  
 2{  
 3  "content":"刘强东方才只是睡觉了,并没有违法"  
 4}  
 5POST doctor\_index/\_search  
 6{  
 7  "query": {  
 8    "match\_phrase": {  
 9      "content": "刘强东"  
10    }  
11  }  
12}

|  |  |
| --- | --- |
| token | position |
| 刘强 | 0 |
| 东方 | 1 |
| 方才 | 2 |
| 方 | 3 |
| 才 | 4 |
| 只是 | 5 |
| 睡觉 | 6 |
| 睡 | 7 |
| 觉了 | 8 |
| 觉 | 9 |
| 了 | 10 |
| 并没有 | 11 |
| 并没 | 12 |
| 并 | 13 |
| 没有 | 14 |
| 有 | 15 |
| 违法 | 16 |
| 法 | 17 |

“刘强东”的分词结果是：

|  |  |
| --- | --- |
| oken | position |
| 刘强 | 0 |
| 东 | 1 |

由于match\_phrase匹配必须position一致，

此例中，由于东方组成了短语，导致结果position不一致，匹配结果检索不到。

让存在的字符都能搜索到

**1**，关于**match\_phrase**的精确度问题

方案一：match\_phrase\_prefix结合slop的方案  
实际开发中，根据应用场景不同，采用不同的分词器。   
如果选用ik，建议使用ik\_max\_word分词，因为：ik\_max\_word的分词结果包含ik\_smart。

* 匹配的时候，如果想尽可能的多检索结果，考虑使用match;
* 如果想尽可能精确的匹配分词结果，考虑使用match\_phrase;
* 如果短语匹配的时候，怕遗漏，考虑使用match\_phrase\_prefix。

但是，事后分析发现：slop设置不论多大、多小，都可能会引入噪音数据，导致结果不准确。

方案二：match\_phrase结合match should关联匹配。  
缺点：依然会引入噪音数据

**2**，逐个字分词和**ik**分词结合的方式

来自阿里云的实践思路

单字分词应用场景——对于像姓名类字段，业务上需要支持完整匹配，又需要支持单字查询。可以配置1个keyword字段（不分词）；

1个text字段（分词），分词器选择Elasticsearch默认分词器standard，按单个汉字切分。

PUT mx\_index  
 2{  
 3  "mappings": {  
 4    "\_doc": {  
 5      "properties": {  
 6        "content": {  
 7          "type": "text",  
 8          "analyzer": "ik\_max\_word",  
 9          "fields": {  
10            "standard": {  
11              "type": "text",  
12              "analyzer": "standard"  
13            },  
14            "keyword": {  
15              "type": "keyword",  
16              "ignore\_above": 256  
17            }  
18          }  
19        }  
20      }  
21    }  
22  }  
23}  
24  
25PUT mx\_index/\_doc/1  
26{  
27  "content":"佟大为老婆生了孩子"  
28}  
29  
30POST mx\_index/\_search  
31{  
32  "query": {  
33    "bool": {  
34      "should": [  
35        {  
36          "match\_phrase": {  
37            "content": "佟大"  
38          }  
39        },  
40        {  
41          "match\_phrase": {  
42            "content.standard": "佟大"  
43          }  
44        }  
45      ]  
46    }  
47  }  
48}

结论

* 1）动态更新词库、词典；
* 2）match\_phrase结合slop就能解决问题。

参考

<https://www.elastic.co/guide/cn/elasticsearch/guide/current/slop.html>

<http://doc.codingdict.com/elasticsearch/245/>

<https://mp.weixin.qq.com/s?__biz=MzI2NDY1MTA3OQ==&mid=2247484287&idx=1&sn=e1f4b24f61d37d556828bbcd211707ac&chksm=eaa82b57dddfa241570730dde38b74a3ff9c36927fd84513b0136e4dc3ee7af18154290a27b2&scene=21#wechat_redirect>

<https://mp.weixin.qq.com/s?__biz=MzI2NDY1MTA3OQ==&mid=2247483946&idx=1&sn=e274a5dc333c0f71593ce2974787d201&chksm=eaa82a02dddfa3146281a52ff8dccc9865efedb8f7d03479a45461fa938bbcb4c8fce8483270&scene=21#wechat_redirect>