**Regexp Query**

regexp允许使用正则表达式进行term查询.注意regexp如果使用不正确，会给服务器带来很严重的性能压力。比如.\*开头的查询，将会匹配所有的倒排索引中的关键字，这几乎相当于全表扫描，会很慢。因此如果可以的话，最好在使用正则前，加上匹配的前缀。在正则中如果使用.\*?或者+都会降低查询的性能。

**注意：是term查询,也就是说这个查询不能跨term。**

举个简单的例子:

GET /\_search

{    "query": {        "regexp":{            "name.first": "s.\*y"

        }

    }

}

正则支持的一些标准的用法：

**搜索关键词的一部分**

如果给定的term是abcde

ab.\* 可以匹配

abcd 不可以匹配

也支持使用^或者$来指定开头或者结尾。

**允许特殊字符**

一些特殊字符是需要转义的，比如:

. ? + \* | { } [ ] ( ) " \ # @ & < >  ~

如果想要搜索某个固定的词，也可以加上双引号。

**匹配任何字符**

.可以匹配任意字符，比如

ab...

a.c.e

这几个都可以匹配abcde

**匹配一个或者多个**

使用+表示匹配一个或者多个字符

a+b+        # matchaa+bb+      # matcha+.+        # matchaa+bbb+     # match

上面这些都可以匹配aaabbb

**匹配零个或者多个**

a\*b\*        # match

a\*b\*c\*      # match

.\*bbb.\*     # match

aaa\*bbb\*    # match

上面这些都可以匹配aaabbb

**匹配另个或者一个**

aaa?bbb?    # matchaaaa?bbbb?  # match.....?.?    # matchaa?bb?      # no match

上面这些都可以匹配aaabbb

**支持匹配次数**

使用{}支持匹配指定的最小值和最大值区间

{5}     # repeat exactly 5 times

{2,5}   # repeat at least twice and at most 5 times

{2,}    # repeat at least twice

比如对于字符串:

a{3}b{3}        # matcha{2,4}b{2,4}    # matcha{2,}b{2,}      # match.{3}.{3}        # matcha{4}b{4}        # no matcha{4,6}b{4,6}    # no matcha{4,}b{4,}      # no match

**捕获组**

对于字符串ababab

(ab)+       # matchab(ab)+     # match(..)+       # match(...)+      # no match(ab)\*       # matchabab(ab)?   # matchab(ab)?     # no match(ab){3}     # match(ab){1,2}   # no match

**选择运算符**

支持或操作的匹配，注意这里默认都是最长匹配的。

aabb|bbaa   # matchaacc|bb     # no matchaa(cc|bb)   # matcha+|b+       # no matcha+b+|b+a+   # matcha+(b|c)+    # match

**字符匹配**

支持在[]中进行字符匹配，^代表非的意思

[abc]   # 'a' or 'b' or 'c'[a-c]   # 'a' or 'b' or 'c'[-abc]  # '-' or 'a' or 'b' or 'c'[abc\-] # '-' or 'a' or 'b' or 'c'[^abc]  # any character except 'a' or 'b' or 'c'[^a-c]  # any character except 'a' or 'b' or 'c'[^-abc]  # any character except '-' or 'a' or 'b' or 'c'[^abc\-] # any character except '-' or 'a' or 'b' or 'c'

其中-代表的范围匹配。

**可选的匹配符**

在正则表达式中也支持一些特殊的操作符，可以使用flags字段控制是否开启。

**Complement**

这个表示正则表示匹配一段字符串，比如ab~cd意思是：a开头，后面是b，然后是一堆非c的字符串，最后以d结尾。比如字符串abcdef

ab~df     # matchab~cf     # matchab~cdef   # no matcha~(cb)def # matcha~(bc)def # no match

**Interval**

interval选项支持数值的范围，比如字符串foo80:

foo<1-100>     # match

foo<01-100>    # match

foo<001-100>   # no match

**Intersection**

使用&可以实现多个匹配的连接,比如字符串aaabbb：

aaa.+&.+bbb     # matchaaa&bbb         # no match

**Any**

使用@，可以匹配任意的字符串

**实践**

首先创建索引：

PUT test然后创建映射：

PUT test/\_mapping/test{  "properties": {    "a": {      "type": "string",      "index":"not\_analyzed"

    },    "b":{      "type":"string"

    }

  }

}

添加一条数据：

PUT test/test/1

{  "a":"a,b,c","b":"a,b,c"}

先来分析一下，a,b,c被默认分析成了什么？

POST test/\_analyze

{  "analyzer": "standard",  "text": "a,b,c"}

返回内容：

{  "tokens": [

    {      "token": "a",      "start\_offset": 0,      "end\_offset": 1,      "type": "<ALPHANUM>",      "position": 0

    },

    {      "token": "b",      "start\_offset": 2,      "end\_offset": 3,      "type": "<ALPHANUM>",      "position": 1

    },

    {      "token": "c",      "start\_offset": 4,      "end\_offset": 5,      "type": "<ALPHANUM>",      "position": 2

    }

  ]

}

然后查询一下：

POST /test/test/\_search?pretty

{  "query":{    "regexp":{        "a": "a.\*b.\*"

    }

  }

}

返回

{  "took": 2,  "timed\_out": false,  "\_shards": {    "total": 5,    "successful": 5,    "failed": 0

  },  "hits": {    "total": 1,    "max\_score": 1,    "hits": [

      {        "\_index": "test",        "\_type": "test",        "\_id": "1",        "\_score": 1,        "\_source": {          "a": "a,b,c",          "b": "a,b,c"

        }

      }

    ]

  }

}

再换成b字段试试：

POST /test/test/\_search?pretty

{

  "query":{    "regexp":{        "b": "a.\*b.\*"

    }

  }

}

返回

{  "took": 1,  "timed\_out": false,  "\_shards": {    "total": 5,    "successful": 5,    "failed": 0

  },  "hits": {    "total": 0,    "max\_score": null,    "hits": []

  }

}

这是为什么呢？

因为整个regexp查询是应用到一个词上的，针对某个词，搜索a.\*b.\*，a字段由于不分词，它的词是整个的a.b.c；b字段经过分词，他的词是a和b和c三个独立的词，因此针对a字段的正则搜索可以查询到结果；但是针对b字段却搜索不到。

归纳起来，还是需要好好理解分词在搜索引擎中的作用才行。