# 排序与相　性

　情况下，返回的　果是按照　　相　性　　行排序的——最相　的文　排在最前。　在本章的后面部分，我　会解　　相　性　意味着什　以及它是如何　算的，不　　我　首先看看 <code>sort</code> 参数以及如何使用它。

## 排序

　了按照相　性来排序，需要将相　性表示　一个数　。在 Elasticsearch 中，　相　性得分 由一个浮点数　行表示，并在搜索　果中通　 `_score` 参数返回，　　排序是 `_score` 降序。

有　，相　性　分　　来　并没有意　。例如，下面的　　返回所有 `user_id` 字段包含 `1` 的　果：

```
GET /_search
{
    "query" : {
        "bool" : {
            "filter" : {
                "term" : {
                    "user_id" : 1
                }
            }
        }
    }
}
```

　里没有一个有意　的分数：因　我　使用的是 filter （　），　表明我　只希望　取匹配 `user_id: 1` 的文　，并没有　　定　些文　的相　性。　　上文　将按照随机　序返回，并且　个文　都会　零分。

**NOTE**

如果　分　零　　造成了困　，　可以使用 `constant_score`　　行替代：

```
GET /_search
{
    "query" : {
        "constant_score" : {
            "filter" : {
                "term" : {
                    "user_id" : 1
                }
            }
        }
    }
}
```

　将　所有文　　用一个恒定分数（　　`1`　　）。它将　行与前述　　相同的　，并且所有的文　将像之前一　随机返回，　些文　只是有了一个分数而不是零分。

## 按照字段的 排序

在 个案例中，通 来 tweets 行排序是有意 的，最新的 tweets 排在最前。 我 可以使用 sort 参数 行 ：

```
GET /_search
{
    "query" : {
        "bool" : {
            "filter" : { "term" : { "user_id" : 1 }}
        }
    },
    "sort": { "date": { "order": "desc" }}
}
```

会注意到 果中的 个不同点：

```
"hits" : {
    "total" :           6,
    "max_score" :       null, ①
    "hits" : [ {
        "_index" :      "us",
        "_type" :       "tweet",
        "_id" :         "14",
        "_score" :      null, ①
        "_source" :     {
             "date":    "2014-09-24",
             ...
        },
        "sort" :        [ 1411516800000 ] ②
    },
    ...
}
```

① `_score` 不被 算, 因 它并没有用于排序。

② `date` 字段的 表示 自 epoch (January 1, 1970 00:00:00 UTC)以来的 秒数，通 `sort` 字段的 行返回。

首先我 在 个 果中有一个新的名 `sort` 的元素，它包含了我 用于排序的 。 在 个案例中，我 按照 `date` 行排序，在内部被索引 自 *epoch* 以来的 秒数 。 long 型数 `1411516800000` 等于日期字符串 `2014-09-24 00:00:00 UTC` 。

其次 `_score` 和 `max_score` 字段都是 `null` 。 算 `_score` 的花 巨大，通常 用于排序； 我 并不根据相 性排序，所以 `_score` 是没有意 的。如果无 如何 都要 算 `_score` ， 可以将 `track_scores` 参数 置 `true` 。

**TIP**

一个 便方法是， 可以指定一个字段用来排序：

```
    "sort": "number_of_children"
```

字段将会 升序排序，而按照 `_score` 的 行降序排序。

## 多 排序

假定我 想要 合使用 `date` 和 `_score` 行 ，并且匹配的 果首先按照日期排序，然后按照相性排序：

```
GET /_search
{
    "query" : {
        "bool" : {
            "must":   { "match": { "tweet": "manage text search" }},
            "filter" : { "term" : { "user_id" : 2 }}
        }
    },
    "sort": [
        { "date":   { "order": "desc" }},
        { "_score": { "order": "desc" }}
    ]
}
```

排序条件的 序是很重要的。 果首先按第一个条件排序， 当 果集的第一个 `sort` 完全相同才会按照第二个条件 行排序，以此 推。

多 排序并不一定包含 `_score` 。 可以根据一些不同的字段 行排序，如地理距 或是脚本 算的特定。

**NOTE**

Query-string 搜索 也支持自定 排序，可以在 字符串中使用 `sort` 参数：

```
GET /_search?sort=date:desc&sort=_score&q=search
```

## 字段多 的排序

一 情形是字段有多个 的排序， 需要 住 些 并没有固有的 序；一个多 的字段 是多个的包装， 个 行排序 ？

于数字或日期， 可以将多 字段 ， 可以通 使用 `min` 、 `max` 、 `avg` 或是 `sum` 排序模式 。例如 可以按照 个 `date` 字段中的最早日期 行排序，通 以下方法：

```
"sort": {
    "dates": {
        "order": "asc",
        "mode":  "min"
    }
}
```

## 字符串排序与多字段

被解析的字符串字段也是多 字段， 但是很少会按照 想要的方式 行排序。如果 想分析一个字符串，如 fine old art ， 包含 3 。我 很可能想要按第一 的字母排序，然后按第二 的字母排序， 如此 ，但是 Elasticsearch 在排序 程中没有 的信息。

可以使用 min 和 max 排序模式（ 是 min ），但是 会 致排序以 art 或是 old ，任何一个都不是所希望的。

了以字符串字段 行排序， 个字段 包含一 ： 整个 not_analyzed 字符串。 但是我 需要 analyzed 字段， 才能以全文 行

一个 的方法是用 方式 同一个字符串 行索引， 将在文 中包括 个字段： analyzed 用于搜索， not_analyzed 用于排序

但是保存相同的字符串 次在 source 字段是浪 空 的。 我 真正想要做的是 一个 _ 字段 但是却用 方式索引它。所有的 _core_field 型 (strings, numbers, Booleans, dates) 接收一个 fields 参数

参数允 化一个 的映射如：

```
"tweet": {
    "type":     "string",
    "analyzer": "english"
}
```

一个多字段映射如：

```
"tweet": { ①
    "type":     "string",
    "analyzer": "english",
    "fields": {
        "raw": { ②
            "type":  "string",
            "index": "not_analyzed"
        }
    }
}
```

① tweet 主字段与之前的一 ：是一个 analyzed 全文字段。

② 新的 tweet.raw 子字段是 not_analyzed.

在，至少只要我　重新索引了我　的数据，使用 tweet 字段用于搜索，tweet.raw 字段用于排序：

```
GET /_search
{
    "query": {
        "match": {
            "tweet": "elasticsearch"
        }
    },
    "sort": "tweet.raw"
}
```

**WARNING**　以全文 analyzed 字段排序会消耗大量的内存。　取更多信息　看 [aggregations-and-analysis] 。

# 什　是相　性?

我　曾　　，　　情况下，返回　果是按相　性倒序排列的。但是什　是相　性？相　性如何　算？

　个文　都有相　性　分，用一个正浮点数字段 _score 来表示 。_score 的　分越高，相　性越高。

　　句会　　个文　生成一个　　score　　字段。　分的　算方式取决于　　型　　不同的　句用于不同的目的：　fuzzy　　会　算与　　的　写相似程度，terms　　会　算到的内容与　　成部分匹配的百分比，但是通常我　　的　_relevance　是我　用来　算全文本字段的　相　于全文本　索　相似程度的算法。

Elasticsearch 的相似度算法被定　　索　　率/反向文　　率，*TF/IDF*，包括以下内容：

索　　率

　索　在　字段出　的　率？出　　率越高，相　性也越高。　字段中出　　5　次要比只出　　1　次的相　性高。

反向文　　率

　　个　索　在索引中出　的　率？　率越高，相　性越低。　索　出　在多数文　中会比出　在少数文　中的　重更低。

字段　度准

　字段的　度是多少？　度越　，相　性越低。　索　出　在一个短的 title 要比同　的　出　在一个　的　content 字段　重更大。

　个　　可以　合使用 TF/IDF 和其他方式，比如短　　中　索　的距　或模糊　　里的　索　相似度。

相　性并不只是全文本　索的　利。也　用于 yes|no 的子句，匹配的子句越多，相　性　分越高。

如果多条　　子句被合并　一条　合　　句，比如　　bool　　，　个　　子句　算得出的　分会被合并到　的相　性　分中。

**TIP**　我　有一　整章着眼于相　性　算和如何　其配合　的需求 [controlling-relevance]。

## 理解 分 准

当 一条 的 句 ，想要理解 _score 究竟是如何 算是比 困 的。Elasticsearch 在个 句中都有一个 explain 参数，将 explain true 就可以得到更 的信息。

```
GET /_search?explain ①
{
   "query"   : { "match" : { "tweet" : "honeymoon" }}
}
```

① explain 参数可以 返回 果添加一个 _score 分的得来依据。

**NOTE**

加一个 explain 参数会 个匹配到的文 生一大堆 外内容，但是花 去理解它是很有意 的。 如果 在看不明白也没 系 — 等 需要的 候再来回一 就行。下面我 来一点点的了解 知 点。

首先，我 看一下普通 返回的元数据：

```
{
    "_index" :      "us",
    "_type" :       "tweet",
    "_id" :         "12",
    "_score" :      0.076713204,
    "_source" :     { ... trimmed ... },
```

里加入了 文 来自于 个 点 个分片上的信息， 我 是比 有 助的，因 率和 文 率是在 个分片中 算出来的，而不是 个索引中：

```
    "_shard" :      1,
    "_node" :       "mzIVYCsqSWCG_M_ZffSs9Q",
```

然后它提供了 _explanation 。 个入口都包含一个 description 、 value 、 details 字段，它分告 算的 型、 算 果和任何我 需要的 算 。

```
"_explanation": { ①
   "description": "weight(tweet:honeymoon in 0)
                   [PerFieldSimilarity], result of:",
   "value":       0.076713204,
   "details": [
      {
         "description": "fieldWeight in 0, product of:",
         "value":       0.076713204,
         "details": [
            {  ②
               "description": "tf(freq=1.0), with freq of:",
               "value":       1,
               "details": [
                  {
                     "description": "termFreq=1.0",
                     "value":       1
                  }
               ]
            },
            { ③
               "description": "idf(docFreq=1, maxDocs=1)",
               "value":       0.30685282
            },
            { ④
               "description": "fieldNorm(doc=0)",
               "value":        0.25,
            }
         ]
      }
   ]
}
```

① honeymoon 相　性　分　算的

②　索　　率

③ 反向文　　率

④ 字段　度准

| | |
|---|---|
| **WARNING** | 出　explain　　果代　是十分昂　的，它只能用作　　工具　。千万不要用于生境。 |

第一部分是　于　算的　　。告　了我　honeymoon 在 tweet 字段中的　索　　率/反向文　　率或TF/IDF，（　里的文　0 是一个内部的 ID，跟我　没有　系，可以忽略。）

然后它提供了　重是如何　算的　　：

　索　　率:

```
  索   `honeymoon` 在   个文   的 `tweet` 字段中的出   次数。
```

反向文     率:

```
  索   `honeymoon` 在索引上所有文   的 `tweet` 字段中出   的次数。
```

字段   度准   :

```
在   个文   中,  `tweet` 字段内容的   度 -- 内容越   ,    越小。
```

   的        句解   也非常      ，但是包含的内容与上面例子大致相同。     通       段信息我   可以了解搜索果是如何   生的。

> **TIP** JSON 形式的 `explain` 描述是   以      的，  但是   成 YAML 会好很多，只需要在参数中加上 `format=yaml` 。

### 理解文   是如何被匹配到的

当       `explain`          加到某一文   上   ，       `explain`       api       会   助   理解   何   个文   会被匹配，更重要的是，一个文      何没有被匹配。

  求路径   `/index/type/id/_explain` ，如下所示：

```
GET /us/tweet/12/_explain
{
   "query" : {
      "bool" : {
         "filter" : { "term" :  { "user_id" : 2           }},
         "must" :  { "match" : { "tweet" :   "honeymoon" }}
      }
   }
}
```

不只是我   之前看到的充分解     ，我      在有了一个 `description` 元素，它将告   我    ：

```
"failure to match filter: cache(user_id:[2 TO 2])"
```

也就是   我   的 `user_id`      子句使   文   不能匹配到。

# Doc Values 介

本章的最后一个      是   于        Elasticsearch        内部的一些   行情况。在   里我   先不介   新的知点，所以我        意   到，Doc Values 是我   需要反   提到的一个重要      。

当 一个字段 行排序 ，Elasticsearch 需要 个匹配到的文 得到相 的 。倒排索引的 索性能是非常快的，但是在字段 排序 却不是理想的 。

- 在搜索的 候，我 能通 搜索 快速得到 果集。
- 当排序的 候，我 需要倒排索引里面某个字段 的集合。 句 ，我 需要 倒置 倒排索引。

倒置 在其他系 中 常被称作 列存 。 上，它将所有 字段的 存 在 数据列中， 使得其 行操作是十分高效的，例如排序。

在 Elasticsearch 中，doc values 就是一 列式存 ， 情况下 个字段的 doc values 都是激活的，doc values 是在索引 建的，当字段索引 ，Elasticsearch 了能 快速 索，会把字段的 加入倒排索引中，同 它也会存 字段的 doc values。

Elasticsearch 中的 doc vaules 常被 用到以下 景：

- 一个字段 行排序
- 一个字段 行聚合
- 某些 ，比如地理位置
- 某些与字段相 的脚本 算

因 文 被序列化到磁 ，我 可以依 操作系 的 助来快速 。当 `working set` 小于 点的可用内存，系 会自 将所有的文 保存在内存中，使得其 写十分高速； 当其大于可用内存，操作系 会自 把 doc values 加 到系 的 存中，从而避免了 jvm 堆内存溢出 常。

我 后会深入 doc values。 在所有 需要知道的是排序 生在索引 建立的平行数据 中。