通　聚合　　常指

significant_terms　（SigTerms）聚合　与其他聚合都不相同。　目前　止我　看到的所有聚合在本上都是　的数学　算。将不同　些　造　相互　合在一起，我　可以　建　的聚合以及数据　表。

significant_terms　有着不同的工作机制。　有些人来　，它甚至看起来有点像机器学　。significant_terms 聚合可以在　数据集中　到一些　常 的指　。

如何解　些　不常　的行　？　些　常的数据指　通常比我　估出　的　次要更　繁，　些　上的　常指　通常象征着数据里的某些有趣信息。

例如，假　我　和跟踪信用　欺　，客　打　来抱怨他　信用　出　常交易，它　的　已　被盗用。　些交易信息只是更　重　的症状。在最近的某些地区，一些商家有意的盗取客　的信用　信息，或者它　自己的信息无意中也被盗取。

我　的任　是　到　危害的共同点　，如果我　有　100　个客　抱怨交易　常，他　很有可能都属于同一个商　，而　家商　有可能就是罪魁　首。

当然，　里面　有一些特例。例如，很多客　在它　近期交易　史　中会有很大的商　如　，我　可以将　排除在外，然而，在最近一些有　的信用　的商家里面也有　。

　是一个 普通的共同 商　的例子。　个人都共享　个商　，无　有没有遭受危害。我　它并不感　趣。

相反，我　有一些很小商　比如街角的一家　店，它　属于　普通但不　常　的情况，只有一　个客有交易　。我　同　可以将　些商　排除，因　所有受到危害的信用　都没有与　些商　生　交易，我　可以肯定它　不是安全漏洞的　任方。

我　真正想要的是　不普通的共同　商　。所有受到危害的信用　都与它　生　交易，但是在未受危害的背景噪声下，它　并不明　。　些商　属于　常，它　比　出　的　率要高。　些不普通的共同商　很有可能就是需要　的。

significant_terms　聚合就是做　些事情。它分析　的数据并通　比正常数据　到可能有　常　次的指　。

暴露的　常指　代表什　依　的　数据。　于信用　数据，我　可能会想　出信用　欺　。　于　商数据，我　可能会想　出未被　的人口信息，从而　行更高效的市　推广。　如果我　正在分析日志，我　可能会　一个服　器会　出比它本　出的更多　常。　significant_terms　的用　不止　些。

significant_terms 演示

因　significant_terms　聚合　是通　分析　信息来工作的，　需要　数据　置一个　它　更有效。也就是　无法通　只索引少量示例数据来展示它。

正因如此，我　准　了一个大　8000　个文　的数据集，并将它的快照保存在一个公共演示　中。可以通　以下　在集群中　原　些数据：

1. 在 elasticsearch.yml 配置文件中　加以下配置，以便将演示　加入到白名　中：

```
repositories.url.allowed_urls: ["http://download.elastic.co/*"]
```

2. 重　Elasticsearch。
3. 　行以下快照命令。（更多使用快照的信息，参　　集群（Backing Up Your Cluster））。

```
PUT /_snapshot/sigterms ①
{
    "type": "url",
    "settings": {
        "url": "http://download.elastic.co/definitiveguide/sigterms_demo/"
    }
}

GET /_snapshot/sigterms/_all ②

POST /_snapshot/sigterms/snapshot/_restore ③

GET /mlmovies,mlratings/_recovery ④
```

① 注　一个新的只　地址　，并指向演示快照。

② （可　）　　内　于快照的　　信息。

③　始　原　程。会在集群中　建　个索引：`mlmovies` 和 `mlratings`。

④ （可　）使用 Recovery API　控　原　程。

NOTE 数据集有 50 MB 会需要一些　　下　。

在本演示中，会看看　MovieLens　里面用　　影的　分。在　MovieLens　里，用　可以推　　影并分，　　其他用　也可以　到新的　影。　　了演示，会基于　入的　影采用 `significant_terms`　　影　行推　。

我　看看示例中的数据，感受一下要　理的内容。本数据集有　个索引，　`mlmovies`　和　`mlratings`。首先　看 `mlmovies`：

```
GET mlmovies/_search ①

{
   "took": 4,
   "timed_out": false,
   "_shards": {...},
   "hits": {
      "total": 10681,
      "max_score": 1,
      "hits": [
         {
            "_index": "mlmovies",
            "_type": "mlmovie",
            "_id": "2",
            "_score": 1,
            "_source": {
               "offset": 2,
               "bytes": 34,
               "title": "Jumanji (1995)"
            }
         },
         ....
```

① 行一个不 条件的搜索，以便能看到一 随机演示文 。

mlmovies 里的 个文 表示一个 影，数据有 个重要字段： 影ID _id 和 影名 title 。可以忽略 offset 和 bytes 。它 是从原始 CSV 文件抽取数据的 程中 生的中 属性。数据集中有 10，681 部影片。

在来看看 mlratings ：

```
GET mlratings/_search

{
   "took": 3,
   "timed_out": false,
   "_shards": {...},
   "hits": {
      "total": 69796,
      "max_score": 1,
      "hits": [
         {
            "_index": "mlratings",
            "_type": "mlrating",
            "_id": "00IC-2jDQFiQkpD6vhbFYA",
            "_score": 1,
            "_source": {
               "offset": 1,
               "bytes": 108,
               "movie": [122,185,231,292,
                  316,329,355,356,362,364,370,377,420,
                  466,480,520,539,586,588,589,594,616
               ],
               "user": 1
            }
         },
         ...
```

里可以看到 个用 的推 信息。 个文 表示一个用 ，用 ID 字段 `user` 来表示， `movie` 字段 一个用 看和推 的影片列表。

基于流行程度推 （Recommending Based on Popularity）

可以采取的首个策略就是基于流行程度向用 推 影片。 于某部影片， 到所有推 它的用 ，然后将他 的推 行聚合并 得推 中最流行的五部。

我 可以很容易的通 一个 `terms` 聚合 以及一些 来表示它，看看 *Talladega Nights*（塔拉 加之夜） 部影片，它是 Will Ferrell 主演的一部 于全国 汽 （NASCAR racing）的喜 。在理想情况下，我 的推 到 似 格的喜 （很有可能也是 Will Ferrell 主演的）。

首先需要 到影片 *Talladega Nights* 的 ID：

```
GET mlmovies/_search
{
  "query": {
    "match": {
      "title": "Talladega Nights"
    }
  }
}

    ...
    "hits": [
     {
        "_index": "mlmovies",
        "_type": "mlmovie",
        "_id": "46970", ①
        "_score": 3.658795,
        "_source": {
           "offset": 9575,
           "bytes": 74,
           "title": "Talladega Nights: The Ballad of Ricky Bobby (2006)"
        }
     },
    ...
```

① *Talladega Nights* 的 ID 是 `46970` 。

有了 ID，可以　　分，再　用 `terms` 聚合从喜　*Talladega Nights* 的用　中　到最流行的影片：

```
GET mlratings/_search
{
  "size" : 0, ①
  "query": {
    "filtered": {
      "filter": {
        "term": {
          "movie": 46970 ②
        }
      }
    }
  },
  "aggs": {
    "most_popular": {
      "terms": {
        "field": "movie", ③
        "size": 6
      }
    }
  }
}
```

① 次 mlratings，将 果内容大小 置 0 因 我 只 聚合的 果感 趣。

② 影片 *Talladega Nights* 的 ID 使用 器。

③ 最后，使用 terms 桶 到最流行的影片。

在 mlratings 索引下搜索，然后 影片 *Talladega Nights* 的 ID 使用 器。由于聚合是 行操作的，它可以有效的 聚合 果从而得到那些只推 *Talladega Nights* 的用 。最后， 行 terms 聚合得到最流行的影片。 求排名最前的六个 果，因 *Talladega Nights* 本身很有可能就是其中一个 果（并不想重 推 它）。

返回 果就像 ：

```
{
...
   "aggregations": {
      "most_popular": {
         "buckets": [
            {
               "key": 46970,
               "key_as_string": "46970",
               "doc_count": 271
            },
            {
               "key": 2571,
               "key_as_string": "2571",
               "doc_count": 197
            },
            {
               "key": 318,
               "key_as_string": "318",
               "doc_count": 196
            },
            {
               "key": 296,
               "key_as_string": "296",
               "doc_count": 183
            },
            {
               "key": 2959,
               "key_as_string": "2959",
               "doc_count": 183
            },
            {
               "key": 260,
               "key_as_string": "260",
               "doc_count": 90
            }
         ]
      }
   }
...
```

通 一个 的 ，将得到的 果 成原始影片名：

```
GET mlmovies/_search
{
  "query": {
    "filtered": {
      "filter": {
        "ids": {
          "values": [2571,318,296,2959,260]
        }
      }
    }
  }
}
```

最后得到以下列表：

1. Matrix, The（ 客帝国）
2. Shawshank Redemption（肖申克的救 ）
3. Pulp Fiction（低俗小 ）
4. Fight Club（搏 部）
5. Star Wars Episode IV: A New Hope（星球大 IV：曙光乍 ）

好 ， 肯定不是一个好的列表！我喜 所有 些影片。但 是：几乎 个人 都喜 它 。 些影片本来就受大 迎，也就是 它 出 在 个人的推 中都会受 迎。 其是一个流行影片的推 列表，而不是和影片 *Talladega Nights* 相 的推 。

可以通 再次 行聚合 松 ，而不需要 影片 *Talladega Nights* 行 。会提供最流行影片的前五名列表：

```
GET mlratings/_search
{
  "size" : 0,
  "aggs": {
    "most_popular": {
      "terms": {
        "field": "movie",
        "size": 5
      }
    }
  }
}
```

返回列表非常相似：

1. Shawshank Redemption（肖申克的救 ）
2. Silence of the Lambs, The（ 的 羊）
3. Pulp Fiction（低俗小 ）

4. Forrest Gump（阿甘正　）
5. Star Wars Episode IV: A New Hope（星球大　IV：曙光乍　）

然，只是　　最流行的影片是不能足以　建一个良好而又具　　能力的推　系　。

基于　　的推　（Recommending Based on Statistics）

在　景已　　定好，使用 significant_terms 。 significant_terms 会分析喜　影片 *Talladega Nights* 的用　　（　　前端　　用　　），并且　定最流行的　影。　　然后　　个用　（　　后端　　用　）　造一个流行影片列表，最后将　者　行比　。

　　常就是与　　背景相比在前景特征　中　度展　的那些影片。理　上　，它　　是一　喜　，因　喜　Will Ferrell 喜　的人　　些影片的　分会比一般人高。

我　　一下：

```
GET mlratings/_search
{
  "size" : 0,
  "query": {
    "filtered": {
      "filter": {
        "term": {
          "movie": 46970
        }
      }
    }
  },
  "aggs": {
    "most_sig": {
      "significant_terms": { ①
        "field": "movie",
        "size": 6
      }
    }
  }
}
```

① 　置几乎一模一　，只是用 significant_terms 替代了 terms 。

正如所　，　　也几乎是一　的。　　出喜　影片 *Talladega Nights* 的用　，他　　成了前景特征用　。　　情况下， significant_terms 会使用整个索引里的数据作　　背景，所以不需要特　的　理。

与 terms　似，　果返回了一　桶，不　有更多的元数据信息：

```
...
   "aggregations": {
      "most_sig": {
         "doc_count": 271, ①
         "buckets": [
            {
               "key": 46970,
               "key_as_string": "46970",
               "doc_count": 271,
               "score": 256.549815498155,
               "bg_count": 271
            },
            {
               "key": 52245, ②
               "key_as_string": "52245",
               "doc_count": 59, ③
               "score": 17.66462367106966,
               "bg_count": 185 ④
            },
            {
               "key": 8641,
               "key_as_string": "8641",
               "doc_count": 107,
               "score": 13.884387742677438,
               "bg_count": 762
            },
            {
               "key": 58156,
               "key_as_string": "58156",
               "doc_count": 17,
               "score": 9.746428133759462,
               "bg_count": 28
            },
            {
               "key": 52973,
               "key_as_string": "52973",
               "doc_count": 95,
               "score": 9.65770100311672,
               "bg_count": 857
            },
            {
               "key": 35836,
               "key_as_string": "35836",
               "doc_count": 128,
               "score": 9.199001116457955,
               "bg_count": 1610
            }
         ]
 ...
```

① doc_count 展 了前景特征 里文 的数量。

② 个桶里面列出了聚合的 （例如，影片的ID）。

③ 桶内文 的数量 doc_count 。

④ 背景文 的数量，表示 在整个 背景里出 的 度。

可以看到， 得的第一个桶是 *Talladega Nights* 。它可以在所有 271 个文 中 到， 并不意外。 我 看下一个桶： 52245 。

个 ID 影片 *Blades of Glory*（ 誉之刃），它是一部 于男子学 滑 的喜 ，也是由 Will Ferrell 主演。可以看到喜 *Talladega Nights* 的用 它的推 是 59 次。 也意味着 21% 的前景特征用 推 了影片 *Blades of Glory* （ `59 / 271 = 0.2177` ）。

形成 比的是， *Blades of Glory* 在整个数据集合中 被推 了 185 次， 只占 0.26% （ `185 / 69796 = 0.00265` ）。因此 *Blades of Glory* 是一个 常：它在喜 *Talladega Nights* 的用 中是 著的共性（注：uncommonly common）。 就 到了一个好的推 ！

如果看完整的列表，它 都是好的喜 推 （其中很多也是由 Will Ferrell 主演）：

1. Blades of Glory（ 誉之刃）
2. Anchorman: The Legend of Ron Burgundy（王牌播音 ）
3. Semi-Pro（半 手）
4. Knocked Up（一夜大肚）
5. 40-Year-Old Virgin, The（四十 的老 男）

只是 significant_terms 它 大的一个示例，一旦 始使用 significant_terms ，可能 到 的情况，我 不想要最流行的，而想要 著的共性（注：uncommonly common）。 个 的聚合可以 示出一些数据里出人意料的 。