23种非常有用的ElasticSearch查询例子(6)

本系列文章将展示ElasticSearch中23种非常有用的查询使用方法。由于篇幅原因，本系列文章分为六篇，本文是此系列的第五篇文章。欢迎关注大数据技术博客微信公共账号:iteblog_hadoop。

Function Score: Field Value Factor

在某些场景下，你可能想对某个特定字段设置一个因子(factor)，并通过这个因子计算某个文档的相关度(relevance score)。这是典型地基于文档(document)的重要性来抬高其相关性的方式。在下面例子中，我们想找到更受欢迎的图书(是通过图书的评论实现的)，并将其权重抬高，这里可以通过使用field_value_factor来实现：

```
////////////////////////////////////////////////////////////////////
 User: 过往记忆
 Date: 2016-10-02
 Time: 22:57
 bolg: https://www.iteblog.com
 本文地址：https://www.iteblog.com/archives/1768.html
 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 过往记忆博客微信公共帐号：iteblog_hadoop
////////////////////////////////////////////////////////////////////
curl POST :9200/iteblog_book_index/book/_search
{
   "query": {
      "function_score": {
         "query": {
            "multi_match" : {
               "query" : "search engine",
               "fields": ["title", "summary"]
            }
         },
         "field_value_factor": {
            "field" : "num_reviews",
            "modifier": "log1p",
```

```
            "factor" : 2
        }
      }
   },
   "_source": ["title", "summary", "publish_date", "num_reviews"]
}
```

[返回结果]

```
{
   "took": 26,
   "timed_out": false,
   "_shards": {
      "total": 3,
      "successful": 3,
      "failed": 0
   },
   "hits": [
      {
         "_index": "bookdb_index",
         "_type": "book",
         "_id": "1",
         "_score": 0.44831306,
         "_source": {
            "summary": "A distibuted real-time search and analytics engine",
            "num_reviews": 20,
            "title": "Elasticsearch: The Definitive Guide",
            "publish_date": "2015-02-07"
         }
      },
      {
         "_index": "bookdb_index",
         "_type": "book",
         "_id": "4",
         "_score": 0.3718407,
         "_source": {
            "summary": "Comprehensive guide to implementing a scalable search engine using A
pache Solr",
            "num_reviews": 23,
            "title": "Solr in Action",
            "publish_date": "2014-04-05"
         }
      },
      {
         "_index": "bookdb_index",
         "_type": "book",
```

```
        "_id": "3",
        "_score": 0.046479136,
        "_source": {
          "summary": "build scalable search applications using Elasticsearch without having to do complex low-level programming or understand advanced data science algorithms",
          "num_reviews": 18,
          "title": "Elasticsearch in Action",
          "publish_date": "2015-12-03"
        }
      },
      {
        "_index": "bookdb_index",
        "_type": "book",
        "_id": "2",
        "_score": 0.041432835,
        "_source": {
          "summary": "organize text using approaches such as full-text search, proper name recognition, clustering, tagging, information extraction, and summarization",
          "num_reviews": 12,
          "title": "Taming Text: How to Find, Organize, and Manipulate It",
          "publish_date": "2013-01-24"
        }
      }
    ]
}
```


Function Score: Decay Functions

在使用Decay Functions之前，我们需要了解Decay Functions的一些基础。Decay Functions主要有三种：分别是linear、exp以及gauss，分别用于操作数字字段(numeric fields)、日期字段(date fields)以及经/纬度的地理点。这三种Decay Functions都接收以下四种参数：

1、origin：中心点，或者是该字段最有可能的值。所有落在中心点的文档的得分(_score)都是1.0；

2、scale：衰减率。指的是一个文档距离origin获得_score的需要减少多少；

3、decay：衰减。指的是一个文档在相对于origin的scale距离应该得到的_score，默认值是0.5；

4、offset：偏移，所有落入-offset

```
curl POST :9200/iteblog_book_index/book/_search
```

```
{
   "query": {
      "function_score": {
         "query": {
            "multi_match" : {
               "query" : "search engine",
               "fields": ["title", "summary"]
            }
         },
         "functions": [
            {
               "exp": {
                  "publish_date" : {
                     "origin": "2014-06-15",
                     "offset": "7d",
                     "scale" : "30d"
                  }
               }
            }
         ],
         "boost_mode" : "replace"
      }
   },
   "_source": ["title", "summary", "publish_date", "num_reviews"]
}
```

[返回结果]

```
{
   "took": 26,
   "timed_out": false,
   "_shards": {
      "total": 3,
      "successful": 3,
      "failed": 0
   },
   "hits": [
      {
         "_index": "bookdb_index",
         "_type": "book",
         "_id": "4",
         "_score": 0.27420625,
         "_source": {
            "summary": "Comprehensive guide to implementing a scalable search engine using A
pache Solr",
            "num_reviews": 23,
```

```
            "title": "Solr in Action",
            "publish_date": "2014-04-05"
         }
      },
      {
         "_index": "bookdb_index",
         "_type": "book",
         "_id": "1",
         "_score": 0.005920768,
         "_source": {
            "summary": "A distibuted real-time search and analytics engine",
            "num_reviews": 20,
            "title": "Elasticsearch: The Definitive Guide",
            "publish_date": "2015-02-07"
         }
      },
      {
         "_index": "bookdb_index",
         "_type": "book",
         "_id": "2",
         "_score": 0.000011564,
         "_source": {
            "summary": "organize text using approaches such as full-text search, proper name re
cognition, clustering, tagging, information extraction, and summarization",
            "num_reviews": 12,
            "title": "Taming Text: How to Find, Organize, and Manipulate It",
            "publish_date": "2013-01-24"
         }
      },
      {
         "_index": "bookdb_index",
         "_type": "book",
         "_id": "3",
         "_score": 0.0000059171475,
         "_source": {
            "summary": "build scalable search applications using Elasticsearch without having to
do complex low-level programming or understand advanced data science algorithms",
            "num_reviews": 18,
            "title": "Elasticsearch in Action",
            "publish_date": "2015-12-03"
         }
      }
   ]
}
```

Function Score: Script Scoring

如果内置的scoring functions满足不了你的需求，我们就可以使用Script Scoring，通过指定一个Groovy script来计算分数。在下面的例子中，我们写了一个脚本首先考虑publish_date，其次再考虑图书的评论数，因为比较新出版的图书可能没有多少评论数，但是我们并不能不考虑它们。计算分数的脚本如下：

```
publish_date = doc['publish_date'].value
num_reviews = doc['num_reviews'].value
if (publish_date > Date.parse('yyyy-MM-dd', threshold).getTime()) {
  my_score = Math.log(2.5 + num_reviews)
} else {
  my_score = Math.log(1 + num_reviews)
}
return my_score
```

然后查询的时候使用script_score 参数：

```
curl POST https://www.iteblog.com:9200/iteblog_book_index/book/_search
{
   "query": {
      "function_score": {
         "query": {
            "multi_match" : {
               "query" : "search engine",
               "fields": ["title", "summary"]
            }
         },
         "functions": [
            {
               "script_score": {
                  "params" : {
                     "threshold": "2015-07-30"
                  },
                  "script": "publish_date = doc['publish_date'].value; num_reviews = doc['num_reviews'].value; if (publish_date > Date.parse('yyyy-MM-
dd', threshold).getTime()) { return log(2.5 + num_reviews) }; return log(1 + num_reviews);"
               }
            }
         ]
      }
```

```
    },
    "_source": ["title", "summary", "publish_date", "num_reviews"]
}
```

[返回结果]

```
{
    "took": 26,
    "timed_out": false,
    "_shards": {
        "total": 3,
        "successful": 3,
        "failed": 0
    },
    "hits": {
        "total": 4,
        "max_score": 0.8463001,
        "hits": [
            {
                "_index": "bookdb_index",
                "_type": "book",
                "_id": "1",
                "_score": 0.8463001,
                "_source": {
                    "summary": "A distibuted real-time search and analytics engine",
                    "num_reviews": 20,
                    "title": "Elasticsearch: The Definitive Guide",
                    "publish_date": "2015-02-07"
                }
            },
            {
                "_index": "bookdb_index",
                "_type": "book",
                "_id": "4",
                "_score": 0.7067348,
                "_source": {
                    "summary": "Comprehensive guide to implementing a scalable search engine usin
g Apache Solr",
                    "num_reviews": 23,
                    "title": "Solr in Action",
                    "publish_date": "2014-04-05"
                }
            },
            {
                "_index": "bookdb_index",
                "_type": "book",
```

```
            "_id": "3",
            "_score": 0.08952084,
            "_source": {
               "summary": "build scalable search applications using Elasticsearch without having to do complex low-level programming or understand advanced data science algorithms",
               "num_reviews": 18,
               "title": "Elasticsearch in Action",
               "publish_date": "2015-12-03"
            }
         },
         {
            "_index": "bookdb_index",
            "_type": "book",
            "_id": "2",
            "_score": 0.07602123,
            "_source": {
               "summary": "organize text using approaches such as full-text search, proper name recognition, clustering, tagging, information extraction, and summarization",
               "num_reviews": 12,
               "title": "Taming Text: How to Find, Organize, and Manipulate It",
               "publish_date": "2013-01-24"
            }
         }
      ]
   }
}
```


注意：为了使用动态的脚本，我们必须先在 config/elasticsearch.yaml 文件中做好相应的配置，具体请参见：https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html。