序言

第 14 节-全文文本分词查询

操作系统	JDK 版本	Elastic Stack 软件版本	虚拟机版本
Centos7-2003	Java 15.x	7.11.1	VMware 15.5.x

说明

- 全文文本分词查询一直是一个非常特殊的领域，在传统数据库非常难实现，在 Elasticsearch 非常容易，且内置了多种分词器，相应的也提供了多种分词检索的方式。
- 分词本身是一种非精确的查询匹配，如果要提高查询精准度，就需要掌握很多参数调整，本课程正是基于此。
- 字段类型必须是 text，其它类型暂时不支持

分词过程与分词之后

- 理解下默认分词器，看看分词之后是什么样的结果

案例练习

```
POST _analyze
{
  "text": ["Hello world ,I am Elastic King"],
  "analyzer": "standard"
}
```

- 分词之后，词库拆解

```
{
```

```
"tokens" : [
  {
    "token" : "hello",
    "start_offset" : 0,
    "end_offset" : 5,
    "type" : "<ALPHANUM>",
    "position" : 0
  },
  {
    "token" : "world",
    "start_offset" : 6,
    "end_offset" : 11,
    "type" : "<ALPHANUM>",
    "position" : 1
  },
  {
    "token" : "i",
    "start_offset" : 13,
    "end_offset" : 14,
    "type" : "<ALPHANUM>",
    "position" : 2
  },
  {
    "token" : "am",
    "start_offset" : 15,
    "end_offset" : 17,
    "type" : "<ALPHANUM>",
    "position" : 3
  },
  {
    "token" : "elastic",
    "start_offset" : 18,
    "end_offset" : 25,
    "type" : "<ALPHANUM>",
    "position" : 4
```

```
    },
    {
      "token" : "king",
      "start_offset" : 26,
      "end_offset" : 30,
      "type" : "<ALPHANUM>",
      "position" : 5
    }
  ]
}
```

全文文本分词检索

准备数据

- 复制一个索引数据，后面的全文文本检索案例基于此。

案例练习

```
DELETE kibana_sample_data_flights_fulltext
POST _reindex
{
  "source": {
    "index": "kibana_sample_data_flights"
  },
  "dest": {
    "index": "kibana_sample_data_flights_fulltext"
  }
}
```

match_all

- 全匹配查询等同于没有限制
- 查询是有分值计算的，默认 1.0

查询参数

- match_all，关键字，全匹配检索，无任何限制条件，默认分值 1.0
- boost，关键字，分值加权，默认 1.0

案例练习

```
#查询 1
GET kibana_sample_data_flights_fulltext/_search
#查询 2，加权
GET kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "match_all": {
      "boost": 10
    }
  }
}
```

match_none

- 反向全匹配查询，不返回任何结果数据

查询参数

- match_none，关键字，反向匹配，等价与不匹配查询

案例练习

```
#查询 1
GET kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "match_none": {}
  }
}
```

match

- Match 是最常用的全文检索语法，其参数也有很多。

查询参数

- match，关键字，匹配关键字，可以是一段文字，也可以一个字

案例练习

- 原始地机场名称关键字检索，注意排序，越相近的越靠前

```
GET kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "match": {
      "Origin": "International Airport"
    }
  }
```

```
}
```

Request 请求参数

- query，查询语法块，查询表达式
- analyzer，分词器，指定分词器
- auto_generate_synonyms_phrase_query，是否生成多种同义词，以供词项检索，取值范围 true/false，默认 true
- fuzziness，模糊容错搜索，与关系型数据库模糊搜索是不一样的概念;纠错模糊查询，容许部分字段分词与查询分词有部分字符纠错
- max_expansions，默认适配选择的词项数据量，默认 50 组
- prefix_length， 模糊容错搜索，指定前缀长度，跳过容错搜索，默认 0，即不全文不跳过
- fuzzy_transpositions，纠错搜索是否容许数据字符对等交换，取值范围 true/false，默认 true
- fuzzy_rewrite，模糊容错搜索选项，可以重写，若启动了容错搜索才会默认启用
- lenient，是否启用数据内容格式化容错，取值范围 true/false，默认 false
- operator，分词直接匹配关系，取值范围 OR/AND，默认 OR
- minimum_should_match，最小匹配词数量设置，默认只要有 1 个就可以
- zero_terms_query，停用词搜索选项，若搜索文本有停用词，则可能搜索不到数据，可通过设置此选项避免，取值范围 none/all，默认 none，设置 all，等同于 match_all

analyzer

- 设定分词器，对于搜索输入的文本进行分词，然后提交搜索

- 案例练习，注意切换不同的分词器 standard/whitespace，对比前后的查询结果与数量

```
GET kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "match": {
      "Origin": {
        "query": "International Airport",
        "analyzer": "standard"
      }
    }
  }
}
```

operator

- 设定分词直接匹配关系，默认是 or，取值范围 or /and，用户控制查询精准度

- 案例练习，注意切换 or/and 关键字，对比前后的数据量

```
GET kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "match": {
      "Origin": {
        "query": "Xianyang International Airport",
        "analyzer": "standard",
        "operator": "and"
      }
    }
```

```
  }
}
```

minimum_should_match

- 设定分词最小匹配度，控制分词数量与查询字段的匹配数量，默认是 1，用于提高查询匹配精准度
- 取值范围支持，整数(1)，百分比(50%+，范围数据(3<90%)
- 案例练习，注意调整数值大小，设置数值：1，2，3，4，设置百分比：10%,50%,75%，对比前后的数据量有结果内容

取值	实例	说明
正整数	2	
负整数	-2	
正百分比	20%	
负百分比	-20%	
单一组合范围	2<20%	
多种组合范围	2<20% 换行符号 3<30%	

```
GET kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "match": {
      "Origin": {
        "query": "International Airport",
        "analyzer": "standard",
        "minimum_should_match": 1
      }
    }
  }
}
```

fuzziness

- 模糊纠错查询，对于文本数据，容许词项有一定的单词拼错，默认查询是关闭状态

- 取值范围，设定 AUTO 或者数字（1，2，3，4，5，....）

- 案例练习，设置 query 的单词，改变其中的字母，让它出错；设置 fuzziness，切换不同的值，对比前后查询结果

```
GET kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "match": {
      "Origin": {
        "query": "Airpory",
        "analyzer": "standard",
        "fuzziness": 1
      }
    }
  }
}
```

match_bool_prefix

- 前缀逻辑匹配查询集成了 match 与 bool 组合查询，前面的词基于分词全匹配，最后一个单词基于前缀匹配方式，

  此处无需过度深入，后面学习 term 查询会深入讲解。

查询参数

- match_bool_prefix，关键字

- 其请求的参数类同 match

案例练习

- 切换查询关键字内容，对比前后数据结果

- 比对 bool-should 组合查询

```
#查询 1
GET kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "match_bool_prefix": {
      "Dest": "Pistarini International Airpor"
    }
  }
}
#查询 2
GET /kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "bool" : {
      "should": [
        { "term": { "Dest": "Pistarini" }},
        { "term": { "Dest": "International" }},
        { "prefix": { "Dest": "Airpor"}}
      ]
    }
  }
}
```

match_phase

- 短语匹配搜索，控制短语的数量与顺序，提高精准度

查询参数

- match_phase，关键字，默契严格按照分词顺序

案例练习

- 对比 2 个查询的结果数量，看看有多大的区别

```
#查询 1，match
GET kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "match": {
      "Origin": {
        "query": "Munich Airport",
        "analyzer": "standard"
      }
    }
  }
}
# 查询 2，短语
GET kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "match_phrase": {
      "Dest": "Munich Airport"
    }
  }
}
```

Request 请求参数

- query，短语查询表达式，输入为短语查询内容

- analyzer，短语查询分词器，输入文本的分词器，默认 standard

- slop，短语词之间间隔，默认 0，泛指可以容许有几个词跳跃

案例练习

- 设置下面的 query 内容与 slop 参数，对比前后的结果数量

```
GET kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "match_phrase": {
      "Dest": {
        "query": "Ministro Pistarini International Airport",
        "analyzer": "standard",
        "slop": 1
      }
    }
  }
}
```

match_phrase_prefix

- 短语前缀查询本质上集成了，短语查询与前缀查询，关系是 and

查询参数

- match_phrase_prefix，关键词
- query，查询表达式
- analyzer，短语查询分词器，输入文本的分词器，默认 standard
- slop，短语词之间间隔，默认 0，泛指可以容许有几个词跳跃

案例练习

- 修改短语查询的最后一个词，对比前后查询的结果内容

```
GET kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "match_phrase_prefix": {
      "Dest": {
        "query": "Ministro Pistarini Internationa",
        "analyzer": "standard"
      }
    }
  }
}
```

multi_match

- 在很多业务场景下，需要同时基于多个字段查询匹配，查询条件一样，如电商领域，商品标题与商品描述。

查询参数

- multi_match，关键字
- query，查询内容表达式，输入查询内容
- type，关键字，匹配类型，默认 best_field，选择最佳的分值字段，与组合有差距
- fields，查询范围字段，可以设置多个字段，也可以支持通配符
- boost ，权重，多字段权重采用特殊符号舍得

案例练习

- 依据起点机场与目的地机场，做分词搜索，对比前后的结果数量

```
#查询 1，多个字段
GET /kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "multi_match": {
      "query": "Ministro Pistarini International Airport",
      "type": "best_fields",
      "fields": ["Dest","Origin"]
    }
  }
}
#查询 2，字段采用通配符
GET /kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "multi_match": {
      "query": "Thunder",
      "fields": ["*Weather"]
    }
  }
}
#查询 3，设置个别字段权重
GET /kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "multi_match": {
      "query": "Thunder",
      "fields": ["OriginWeather","DestWeather^2"]
```

```
    }
  }
}
```

type

- 多字段匹配查询，提供了多种匹配类型，如下
- best_fields，多字段中选择分值最高的字段，默认匹配类型
- most_fields，多字段分值累计和
- cross_fields，多字段查询时，部分分词在第一个字段里，其它的分词在另外的字段里
- phrase，短语匹配，等同 match_phase
- phrase_prefix，短语前缀匹配，等同 match_phase_prefix
- bool_prefix，全文匹配逻辑前缀，等同 match_bool_prefix
- 案例练习，切换不同的类型（best_fields/most_fields），测试对比前后的分值与结果数量

```
GET /kibana_sample_data_flights_fulltext/_search
{
  "track_total_hits": true,
  "query": {
    "multi_match": {
      "query": "Ministro Pistarini International Airport",
      "type": "most_fields",
      "fields": ["Dest","Origin"]
    }
  }
}
```

intervals

- 间隔查询是全文分词非常高级的查询能力，容许控制输入分词查询与内容之间的间隔。支持了多种间隔类型机制。

查询语法

- intervals，关键字，间隔查询入口

案例练习

```
POST /kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "intervals": {
      ...各种语法糖
    }
  }
}
```

match 间隔匹配查询

- match，关键字，间隔查询的全文分词方式，等同前面的 match 查询
- query，关键字，查询输入的内容
- max_gaps，关键字，容许中间间隔最大的词数量，默认-1，不限制
- ordered，关键字，查询的内容是否必须符合顺序，取值 true/false，默认 false
- analyzer，关键字，分词器

- filter，关键字，二级查询过滤器，支持多种过滤类型

- use_field，自定义字段类型，

filter 参数说明

- 二级查询过滤器，支持多种过滤类型，详细见下面表格

filter 类型	说明	
after	query 查询在此之后执行	
before	query 查询在此之前执行	
contained_by	包含此执行条件之内的结果	
containing	包含此执行条件	
not_contained_by	不在此执行结果之内	
not_containing	不包含此条件	
not_overlapping	不重叠条件	
overlapping		重叠条件
script	基于 painless 脚本限制	

案例练习

- 搜索 kibana 日志数据，关键字 “elasticsearch Firefox”之间，不包含"GET“关键字，最大的间隔不超过 100；

  设置不同的参数，比对前后的数据内容与结果数量；

```
POST /kibana_sample_data_logs/_search
{
  "track_total_hits": true,
  "query": {
    "intervals": {
      "message": {
        "match": {
          "ordered": true,
          "query": "elasticsearch Firefox",
          "analyzer": "standard",
          "max_gaps": 100,
          "filter": {
            "not_containing": {
              "match": {
                "query": "GET"
```

```
            }
          }
        }
      }
    }
  }
}
```

prefix 间隔前缀查询

- 类同 prefix 查询，建议参考 match 间隔查询
- prefix，关键字参数

wildcard 间隔模糊查询

- 类同 prefixrefix 查询，建议参考 match 间隔查询
- pattern，关键字参数

fuzzy 间隔分词容错查询

- 类同 prefix 查询，建议参考 match 间隔查询
- term，关键字

all_of 多种间隔模式组合

- 组合多种查询模式，必须满足多个条件，建议参考 match 间隔查询
- intervals，关键字

any_of 多种间隔模式组合

- 组合多种查询模式，从其中选择一个，建议参考 match 间隔查询

filter 多种间隔模式组合

- 过滤组合查询模式，从其中选择一个，建议参考 match 间隔查询

query_string

- 查询字符串是一种原始的全文查询方式，相对 DSL 查询表达能力弱一些，但也表现的很轻量。

查询语法

查询参数

- query_string，查询语法关键字

案例练习

```
POST /kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "query_string": {
      "default_field": "Dest",
      "query": "(Phoenix) OR (Ministro)"
```

```
    }
  }
}
```

Request body 请求参数

- query_string，关键字，查询字符串入口
- default_field，关键字，选择一个字段作为查询对象
- query，关键字，构建查询表达式
- analyzer，关键字，查询分词器
- boost，关键字，分值加权
- default_operator，关键字，操作符，OR/AND，默认 OR
- fields，关键字，查询字段列表，设置多个查询字段
- minimum_should_match，关键字，最小匹配度
- fuzziness，关键字，分词单字符容错纠错设置

查询表达式

查询字符串方式，有多种查询表达式，以下列举几种

字段名查询

- 查询表达式中带有字段名字

- 案例练习，目的机场包含 2 个机场名字其中之一，

```
POST /kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "query_string": {
      "query": "Dest:(Phoenix OR Ministro)"
    }
  }
}
```

数值范围查询

- 查询表达式带有数值的范围

- 案例练习，查询延迟飞行时间范围，在 10～100 之间的航班

```
POST /kibana_sample_data_flights_fulltext/_search
{
  "query": {
    "query_string": {
      "query": "FlightDelayMin:[10 TO 100]"
    }
  }
}
```

simple_query_string

- 类同 Query string 查询，容错性更好一点，这里不做过多的阐述

Url 全文检索

类同 Query String，但表现方式基于 Url

url 查询表达式

- q，关键字，查询入口
- 查询表达式，类同 Query String 的查询表达式，注意符合编码

案例练习

- 查询目的集成是 Manchester ，并且起点机场是 Edmonton 的飞行航班信息

#查询 1，基于 q 关键字，查询表达式支持 字段名

GET kibana_sample_data_flights_fulltext/_search?q=(Dest:Manchester) AND (Origin:Edmonton)

查询性能分析评估

profile

- DSL 查询语句分析，分析查询耗时
- profile=true，查看 DSL 执行耗时

案例练习

```
GET kibana_sample_data_flights/_search
{
  "profile": "true",
  "track_total_hits": true
}
```

explain

- 分词查询分值计算，便于评估分值计算规则，从而评估分值计算效率

查询语法

```
GET /<index>/_explain/<id>
POST /<index>/_explain/<id>
```

案例练习

```
POST kibana_sample_data_flights_fulltext/_explain/wI5FzHUBeOqkuB45fSPn
{
  "query": {
    "match": {
      "Dest": "International Airport"
    }
  }
}
```

Full Text Search 源码解读

- 全文检索查询过程本质是走的 SEARCH，前部分源码类同，仅仅后面的执行构建源码不一致

Class

Match 关联类

MatchQueryBuilder

- Match 查询条件建造类

```
package org.elasticsearch.index.query;
/**
 * Match query is a query that analyzes the text and constructs a query as the
 * result of the analysis.
```

```
 */
public class MatchQueryBuilder extends AbstractQueryBuilder<MatchQueryBuilder> {
        public void doXContent(...)
        protected Query doToQuery(...)
}
```

MatchQuery

- MatchQuery 逻辑类

```
package org.elasticsearch.index.search;
public class MatchQuery {
          /*Lucene 类*/
          class MatchQueryBuilder extends QueryBuilder {
                    private Query createFieldQuery(...)
           }
}
```

QueryParserBase

- Query 语法转换类

```
package org.apache.lucene.queryparser.classic;
/** This class is overridden by QueryParser in QueryParser.jj
 * and acts to separate the majority of the Java code from the .jj grammar file.
 */
public abstract class QueryParserBase extends QueryBuilder implements
CommonQueryParserConfiguration {

}
```

QueryBuilder

- 底层，Lucene 查询构建类

```
package org.apache.lucene.util;
public class QueryBuilder {
        protected Query createFieldQuery(...)
```

```
}
```

MatchAllQueryBuilder

- 详细内容请自行展开

MatchNoneQueryBuilder

- 详细内容请自行展开

MultiMatchQueryBuilder

- 详细内容请自行展开

MatchPhraseQueryBuilder

- 详细内容请自行展开

MatchBoolPrefixQueryBuilder

- 详细内容请自行展开

案例

Match 测试案例

- 设置断点， 提交 Match 查询，跟踪调试执行过程

参考文献

全文文本检索

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/full-text-queries.html

最小匹配度

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/query-dsl-minimum-should-match.html

fuzziness 取值范围

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/common-options.html#fuzziness

explain 分值计算分析器

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/search-explain.html

profile 性能优化器

https://www.elastic.co/guide/en/elasticsearch/reference/7.11/search-profile.html

fuzziness 计算算法

https://en.wikipedia.org/wiki/Levenshtein_distance

关于我们

讲师大咖

李猛 Elastic King

1. Elastic Stack 国内顶尖实战专家
2. ELastic Stack 技术社区分享嘉宾
3. 前某大型物流公司大数据架构师
4. 国内首批 Elastic 官方认证工程师 21 人之一
5. 多个 MVP（大数据领域）

2012 年接触 Elasticsearch，对 Elastic Stack 技术栈开发、架构、运维、源码、算法等方面有深入实战，主导过 PB 级以上大规模集群；负责过多种 Elastic Stack 项目，包括大数据领域，机器学习领域，业务系统领域，日志析领域，监控领域等 服务过多家企业，提供 Elastic Stack 咨询培训以及调优实施 多次在 Elastic Stack 技术大会/技术社区分享，发表过多多篇实战干货文章； 十余年技术实战从业经验，擅长大数据多种技术混合，系统架构领域。