Elasticsearch教程(三)

一. 分词器安装

1.1 ik分词器

1.1.1 下载

> 下载地址： https://github.com/medcl/elasticsearch-analysis-ik/releases

1.1.2 安装

> IK分词器在任何操作系统下安装步骤均一样: 在ES的家目录下的 `plugins` 目录下创建名为 `ik` 的文件夹，然后将下载后的 zip 包拷贝到 `ik` 解压即可

> IK分词器提供了两种分词方式：

分词器名称	说明
ik_smart	会做最粗粒度的拆分，比如会将“中华人民共和国国歌”拆分为“中华人民共和国,国歌”，适合 Phrase 查询
ik_max_word	会将文本做最细粒度的拆分，比如会将“中华人民共和国国歌”拆分为“中华人民共和国,中华人民,中华,华人,人民共和国,人民,人,民,共和国,共和,和,国国,国歌”，会穷尽各种可能的组合，适合 Term Query；

1.1.3 验证

> 使用 `ik_smart` 分词器

```
GET _analyze
{
  "analyzer": "ik_smart",
  "text": ["人终究是孤独的，每个人的人生都有别人参与，却都要自己完成。现在已经学会面对和接受孤独，即使伤疤没磨练的足够硬实，也不再依赖别人给的铠甲"]
}
```

> 使用 `ik_max_word` 分词器

```
GET _analyze
{
  "analyzer": "ik_max_word",
  "text": ["人终究是孤独的，每个人的人生都有别人参与，却都要自己完成。现在已经学会面对和接受孤独，即使伤疤没磨练的足够硬实，也不再依赖别人给的铠甲"]
}
```

1.1.4 自定义词库

> 在很多的时候，业务上的一些词库极有可能不在IK分词器的词库中，需要去定制属于我们自己的词库。例如下面的例子中，`正井猫` 、`up主` 被切分为一个个的字，我们希望这两个词语是不被拆分；另外 `的` 作为中文的停顿词，也不希望出现在分词中，所以我们需要自定义词库和停顿词词库。

```
GET _analyze
{
  "analyzer": "ik_max_word",
  "text": ["请关注正井猫up主，你们的支持是我肝下去的动力"]
}
```

IK分词器的结果

> 进入到 `$ES_HOME/plugins/ik/config` 目录下，创建 `custom` 目录，在目录下创建 `mydic.dic` 、`ext_stopword.dic` 文件。

> 在 `mydic.dic` 文件中添加两行内容：

```
正井猫
up主
```

> 在 `ext_stopword.dic` 中添加一行内容:

```
的
```

> 最后修改 `$ES_HOME/plugins/ik/config/IKAnalyzer.cfg.xml` 文件，内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>IK Analyzer 扩展配置</comment>
  <!--用户可以在这里配置自己的扩展字典 -->
  <entry key="ext_dict">custom/mydic.dic</entry>
  <!--用户可以在这里配置自己的扩展停止词字典-->
  <entry key="ext_stopwords">custom/ext_stopword.dic</entry>
  <!--用户可以在这里配置远程扩展字典 -->
  <!-- <entry key="remote_ext_dict">words_location</entry> -->
  <!--用户可以在这里配置远程扩展停止词字典-->
  <!-- <entry key="remote_ext_stopwords">words_location</entry> -->
</properties>
```

> **重启elasticsearch**， 重新执行如上的命令，结果如下:

自定义词库后再次分词的结果

```
GET _analyze
{
  "analyzer": "ik_smart",
  "text": ["关注正井猫up主，你们的支持是我肝下去的动力"]
}
```

```
{
  "tokens" : [
    {
      "token" : "关注",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "CN_WORD",
      "position" : 0
    },
    {
      "token" : "正井猫",
      "start_offset" : 2,
      "end_offset" : 5,
      "type" : "CN_WORD",
      "position" : 1
    },
    {
      "token" : "up主",
      "start_offset" : 5,
      "end_offset" : 8,
      "type" : "CN_WORD",
      "position" : 2
    },
```

将"正井猫"、"up主"添加到词库后，这两个词语就不会被切割。

1.2 pinyin分词器

1.2.1 下载

> 下载地址: https://github.com/medcl/elasticsearch-analysis-pinyin/releases

1.2.2 安装

> pinyin 分词器在任何操作系统下安装步骤均一样: 在ES的家目录下的 `plugins` 目录下创建名为 `pinyin` 的文件夹，然后将下载后的 zip 包拷贝到 `pinyin` 解压即可

1.2.3 验证

> 执行如下命令:

```
GET _analyze
{
  "analyzer": "pinyin",
  "text": "正井猫"
}
```

pinyin分词器的结果

```
GET _analyze
{
  "analyzer": "pinyin",
  "text": "正井猫"
}
```

```
{
  "tokens" : [
    {
      "token" : "zheng",
      "start_offset" : 0,
      "end_offset" : 0,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "zjm",
      "start_offset" : 0,
      "end_offset" : 0,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "jing",
      "start_offset" : 0,
      "end_offset" : 0,
      "type" : "word",
      "position" : 1
    },
    {
      "token" : "mao",
      "start_offset" : 0,
      "end_offset" : 0,
      "type" : "word",
      "position" : 2
    }
  ]
}
```

1.3 自定义分词器以及应用

> 对于 `<p>刘德华</p>` ，现在想要得到如下的分词结果:

```
{
  "tokens" : [
    {
```

```
      "token" : "刘德华",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "liudehua",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "ldh",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "word",
      "position" : 0
    }
  ]
}
```

1.3.1 设置分词器

```
PUT test
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_analyzer": {
          "char_filter": ["html_strip"],
          "tokenizer": "keyword",
          "filter": "my_pinyin_filter"
        }
      },
      "filter": {
        "my_pinyin_filter": {
          "type": "pinyin",
          "keep_first_letter": true,
          "keep_full_pinyin": false,
          "keep_joined_full_pinyin": true,
          "keep_original": true,
          "keep_none_chinese": false,
          "keep_none_chinese_in_joined_full_pinyin": true
        }
      }
    }
  }
}
```

1.3.2 验证分词器效果

```
GET test/_analyze
{
  "analyzer": "my_analyzer",
  "text": ["刘德华"]
}
```

自定义分词的执行结果

```
PUT test
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_analyzer": {
          "char_filter": ["html_strip"],
          "tokenizer": "keyword",
          "filter": "my_pinyin_filter"
        }
      },
      "filter": {
        "my_pinyin_filter": {
          "type": "pinyin",
          "keep_first_letter": true,
          "keep_full_pinyin": false,
          "keep_joined_full_pinyin": true,
          "keep_original": true,
          "keep_none_chinese": false,
          "keep_none_chinese_in_joined_full_pinyin": true
        }
      }
    }
  }
}

GET test/_analyze
{
  "analyzer": "my_analyzer",
  "text": ["刘德华"]
}
```

```
{
  "tokens" : [
    {
      "token" : "刘德华",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "liudehua",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "ldh",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "word",
      "position" : 0
    }
  ]
}
```

1.3.3 为属性添加分词器

> 设定 `mappings` 信息，指定索引 `test` 的 `name` 属性的 `analyzer` 自定义的分词器。

```
PUT test/_mapping
{
  "properties": {
    "name": {
      "type": "completion",
      "analyzer": "my_analyzer"
    }
  }
}
```

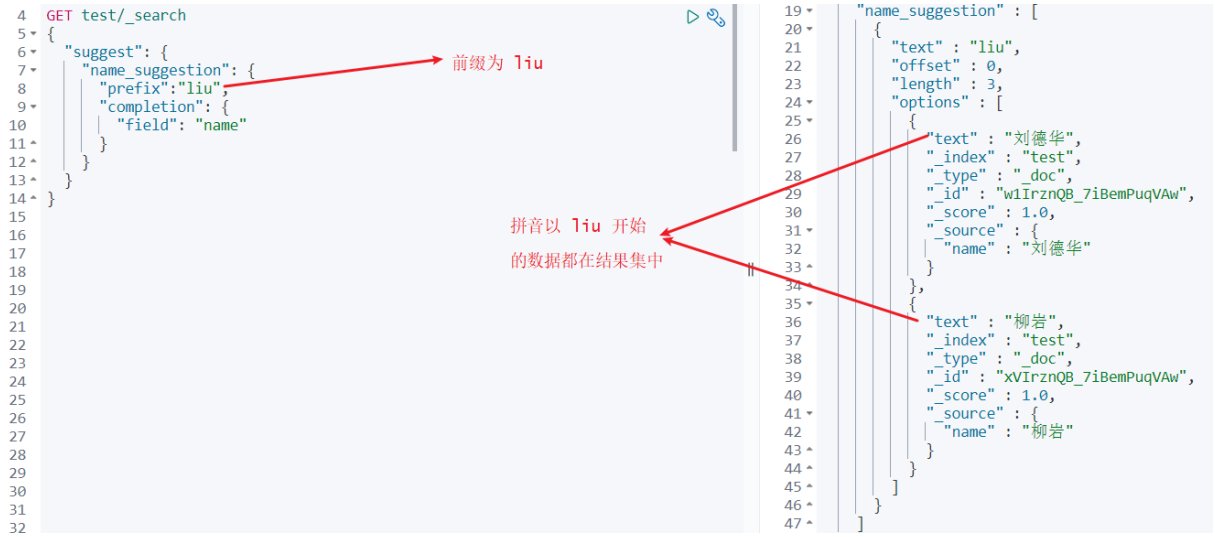
1.3.4 结果验证

> 执行如下命令添加数据

```
POST test/_bulk
{"index": {}}
{"name": "刘德华"}
{"index": {}}
{"name": "张学友"}
{"index": {}}
{"name": "柳岩"}
```

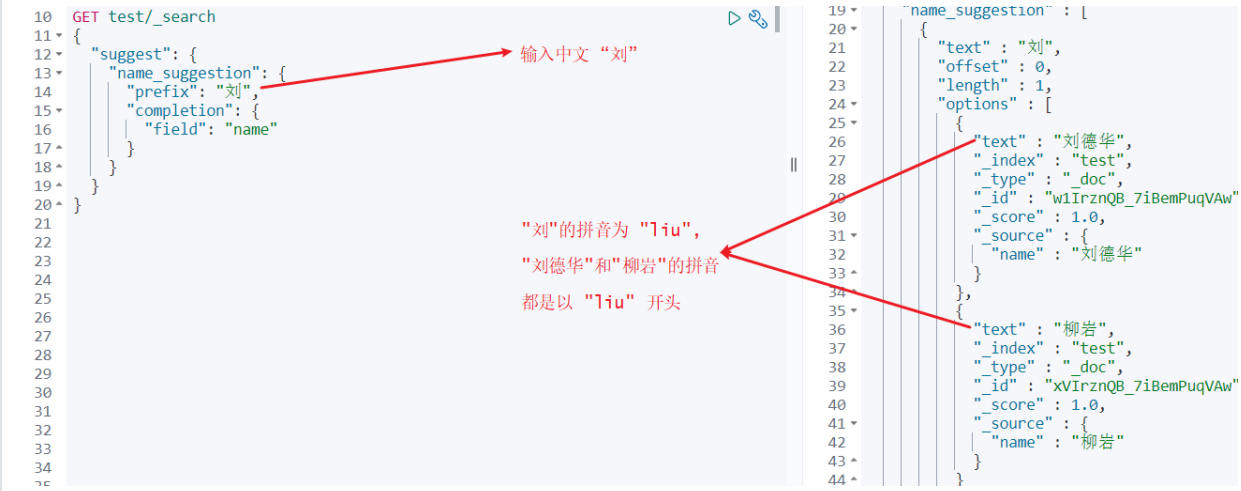
执行前缀建议语句

前缀建议案例(一)



前缀建议案例(二)

前缀建议案例(三)



> **通过如上最后一个结果大家仔细去理解《Elasticsearch教程(一)》中，第5节的开始标红的那句话。**

二. MySQL数据导入到ES

> 将MySQL的初始化数据导入到ES的方式可以通过程序的方式和工具的方式。本教程使用 `Logstash` 来初始化导入。首先将 MySQL 的驱动包拷贝到 `$logStash/logstash-core/lib/jars/` 目录下；在 `$logstash/config/` 目录下创建名为 `logstash-mysql-news.conf` 的文件，文件内容如下：

```
input {
  jdbc {
    jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://localhost:3306/es?
useSSL=false&serverTimezone=UTC"
    jdbc_user => root
    jdbc_password => "123456"
    #启用追踪，如果为true，则需要指定tracking_column
    use_column_value => true
    #指定追踪的字段，
    tracking_column => id
    #追踪字段的类型，目前只有数字(numeric)和时间类型(timestamp)，默认是数字类型
    tracking_column_type => "numeric"
    #记录最后一次运行的结果
    record_last_run => true
    #上面运行结果的保存位置
    last_run_metadata_path => "mysql-position.txt"
    statement => "SELECT * FROM news where id > :sql_last_value"
    schedule => "* * * * * *"
  }
}

filter {
```

```
  mutate {
    split => { "tags" => ","}
  }
}
output {
  elasticsearch {
    document_id => "%{id}"
    document_type => "_doc"
    index => "news"
    hosts => ["http://localhost:9200"]
  }
  stdout{
    codec => rubydebug
  }
}
```