

法律声明

□ 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

■ 微信公众号：大数据分析挖掘

■ 新浪微博：ChinaHadoop



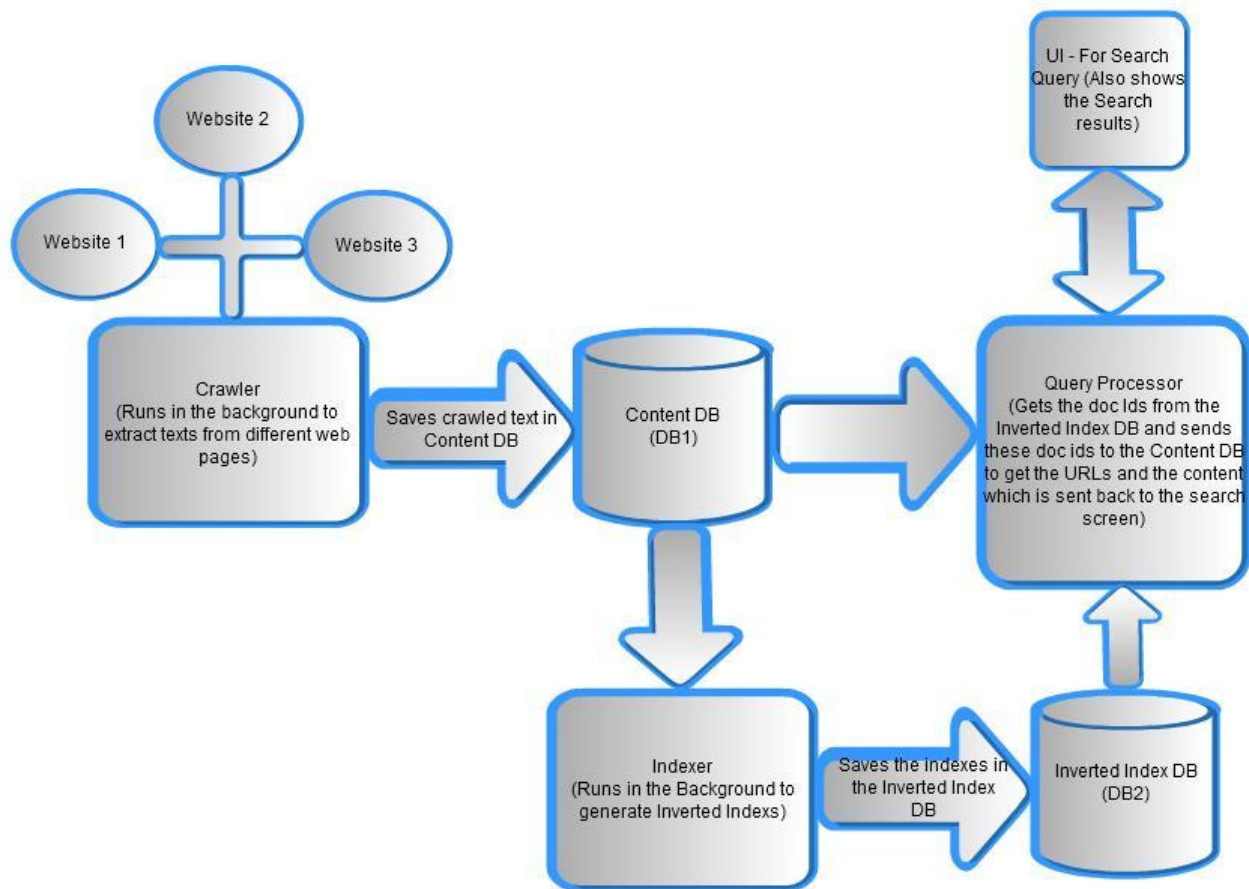
分布式爬虫

大纲

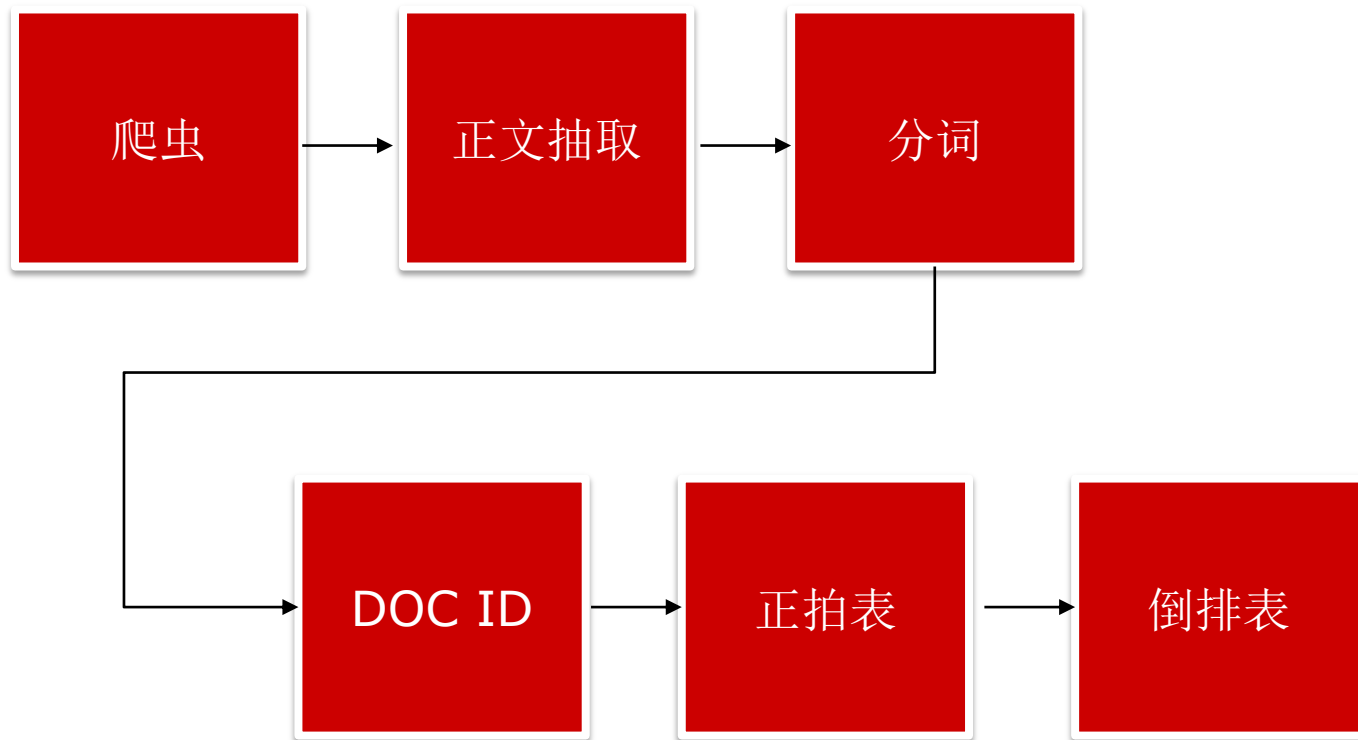
- 倒排索引
- 文本匹配及排序
- Elastic Search

倒排索引

核心-倒排表



处理过程



正排索引

- LocalId: 文档的局部编号
- WordId: 文档分词后的编号
- nHits: 某个索引词出现的次数
- HitList 变长字段: 某个索引词在文档出现的位置

倒排索引

- 词典：不同索引词组成的索引表

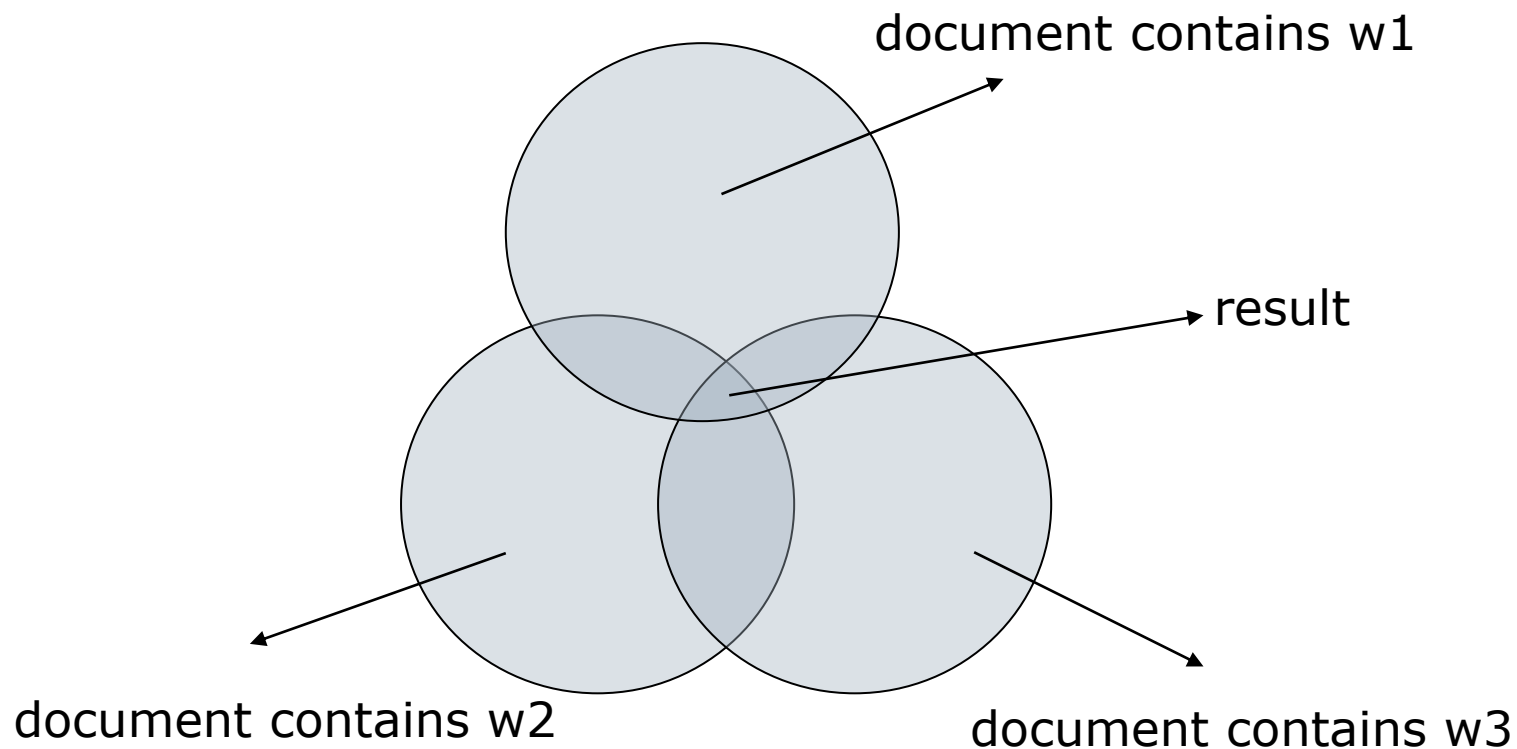
WordID	nDocs	Offset	DocId	NHits	HitList
dog	208	120	Doc1	8	12,35
cat	13	175	Doc2	13	175
...	Doc1		
			Doc2		
			Doc3		
			Doc4		
		

- 记录表：每个索引词出现过的文档集合以及命中位置

查询系统

Boolean 模型

单纯地查询每个检索词，最后把检索词的结果取交集



词与词的关联

在提取特征词的时候，我们可以用 **TfIdf** 来找出特征词，同时**TfIdf** 也可以用来表述一个词 w_i 对于一篇文章 d_j 的重要性 $w_{i,j}$

Boolean的检索方式，词与词之间被看成是独立的，而实际上，搜索词之间的组合，是具备一定关联，关联定义

$$C_{u,v} = \sum_{d_j} w_u \times w_v$$

标准化网页长度

一般来说，一篇网页的文本越长，它被随机搜索词命中的几率也就越大。因此我们需要考虑把网页的长度进行标准化，一般来说，网页文本长度有以下的方式来表达：

- **Size in Bytes:** 网页文本流的长度
- **Size in Words:** 网页字词数量
- 向量的模： $|d_i| = \sqrt{\sum_{i=0}^n w_{i,j}^2}$ 其中 $w_{i,j}$ 是每个词在文档 j 的权重

向量模型

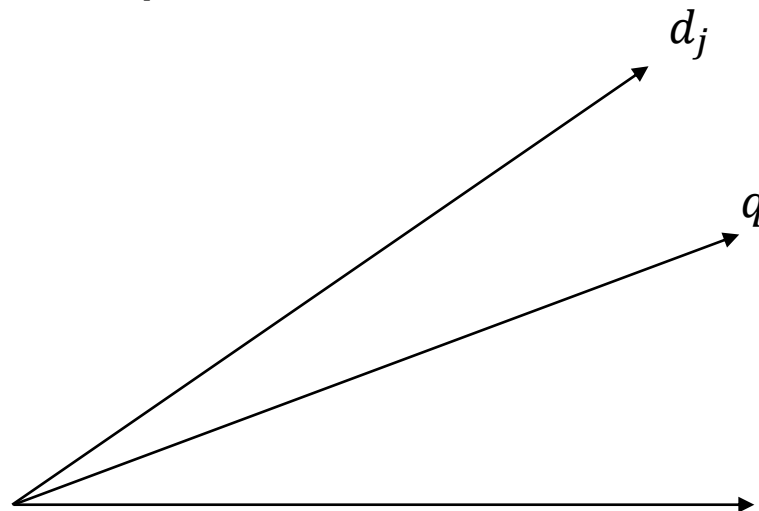
假设一个搜索系统，全部的搜索词数量为 t ，那么搜索词与一篇文档，可以表达为

$$d_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j})$$

$$q = (w_{1,q}, w_{2,q}, \dots, w_{t,q})$$

相似度可以表示为：

$$\text{sim}(d_j, q) = \frac{d_j \cdot q}{\|d_j\| \times \|q\|}$$



概率模型

假设对于一个搜索，相关的文档的全集为 R ，不相关的文档的集合为 R' ，于是我们可以得到一篇文档与查询总的相关度为：

$$\text{sim}(d_j, q) = \frac{P(R|d_j, q)}{P(R'|d_j, q)}$$

我们可以得到这样的一个相关性：

$$\text{sim}(d_j, q) \sim \sum_{k_i \in q \cap k_i \in d_j} \log \left(\frac{p_{iR}}{1 - p_{iR}} \right) + \log \left(\frac{1 - q_{iR}}{q_{iR}} \right)$$

其中 $p_{iR} = P(k_i|R, q)$, $q_{iR} = P(k_i|R', q)$

从集合 R 中一个随机选择一个文档，第 i 个索引词 k_i 出现和不出现在这个文档的概率，就是 p_{iR} 及 q_{iR}

概率模型

$$\text{sim}(d_j, q) \sim \sum_{k_i \in q \cap k_i \in d_j} \log \left(\frac{p_{iR}}{1 - p_{iR}} \right) + \log \left(\frac{1 - q_{iR}}{q_{iR}} \right)$$

集合 R 在最开始不知道的，我们可以这样来推导

	相关搜索	不相关搜索	全部
文档包含 k_i	r_i	$n_i - r_i$	n_i
文档不包含 k_i	$R - r_i$	$N - n_i - (R - r_i)$	$N - n_i$
全部文档	R	$N - R$	N

把上面的关系带入，最后我们可以推导出：

$$\text{sim}(d_j, q) \sim \sum_{k_i \in q \cap k_i \in d_j} \log \left(\frac{(r_i + 0.5)(N - n_i - R + r_i + 0.5)}{(R - r_i + 0.5)(n_i - r_i + 0.5)} \right)$$

概率模型

我们并不知道 R 和 r_i ，因此假设 $R = r_i = 0$ ，得到第一轮的结果：

$$\text{sim}(d_j, q) \sim \sum_{k_i \in q \cap k_i \in d_j} \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$

我们可以把计算的结果，从文档里取出来一个子序列，然后估算得到 R ， r_i ，重新带入到前面的公式，反复地进行计算

几种模型的简单比较

- **Boolean Model:** 比较简单粗暴，结合词的权重能得到不错的结果；如果一篇文章漏掉了一个搜索词就不能被检索到
- **Vector Model:** 容错性能较好，对于比较多的搜索词，查询结果比较好；反过来对于很长的文章，精度不高，对于1、2个词的搜索，效果不好
- **Probability Model:** 理论上结果是最优的，文档按照相关度降序排列，但是实践中，初始值的估算会带来比较大的误差，并没有利用词频以及标准化文档长度

模糊匹配

对于穆罕默德、默罕默德、穆罕穆德，正确写法应该是 穆罕默德，如何确保不管搜索哪个词，都能得到“穆罕穆德”的相关结果？

准备过程

先把所有检索词的拼音记录下来，并保存2个拼音的列表，一个是拼音首字母组合表，第二个是拼音的全拼列表，同样形成拼音->词语的倒排表

词	拼音	首字母
穆罕穆德	muhanmude	mhmd
孔子	kongzi	kz
爱因斯坦	aiyinsitan	ayst

首字母	词
ays	安阳市
ayst	爱因斯坦

拼音	词
muhanmode	穆罕默德
mohanmodi	穆罕默迪

搜索过程

1. 用户开始输入的时候，把完整词、拼音首字母与完整拼音都提取出来，分别进入到索引系统查询
2. 查询的结果，根据匹配完整度和候选词热度综合来排序，一般如果有完整匹配，优先完整匹配的词，然后依次是拼音匹配、热度从高到底
3. 精准匹配失败，意味着搜索词不存在，开始启用纠错功能，纠错的时候，假设搜索词的长度是正确的，只是出现了错别字，因此分别过滤掉一个字进行搜索

日志系统

- 用户信息（ID）
- 查询信息：查询字符串、时间、终端类型、前面的若干个查询词
- 点击信息：点击的URL、时间、点击URL在结果集中的位置
- 结果展现：所搜索结果、广告结果、广告点击

作用

- 用户偏好分析及用户画像
- 热点词排序
- 输入Suggestion及纠错
- 查询结果缓存一句：10%的查询词占据总搜索量的50%
- 类别识别：Harry Potter Movie/Book 统计来对 Harry Potter 自动分类
- 搜索结果优化：根据用户点击来调整网站排名，一般排名靠前的网页本身有比较高的点击率，因此统计分析的时候要把结果进行修正
- 查询词相关性分析：单位时间内连续几个查询词和点击具备相关性

Elastic Search

简介

- 基于 Apache Lucene, Java 开发的搜索引擎
- 开源项目
- 支持分布式部署
- 基于 RESTful 接口的调用方式
- 开源网址: <https://github.com/elastic/elasticsearch>
- 支持集群, 支持分片和复制

<https://www.elastic.co/>

基本术语

- 集群：一个集群就是由一个或多个节点组织在一起，它们共同持有你整个的数据，并一起提供索引和搜索功能
- 节点：一个节点是你集群中的一个服务器，作为集群的一部分，它存储你的数据，参与集群的索引和搜索功能
- 索引：一个索引就是一个拥有几分相似特征的文档的集合
- 类型：在一个索引中，你可以定义一种或多种类型。一个类型是你的索引的一个逻辑上的分类/分区
- 文档：一个文档是一个可被索引的基础信息单元，比如一篇网页可以作为一个文档被加入到系统中

<https://www.elastic.co/>

安装与启动

下载:

<https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.3.0.zip>

启动:

Run bin/elasticsearch (or bin\elasticsearch.bat on Windows)

访问:

<http://localhost:9200/>

<https://www.elastic.co/>

安装 IK 分词器

下载:

open source: <https://github.com/medcl/elasticsearch-analysis-ik>

jar: <https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v5.3.0/elasticsearch-analysis-ik-5.3.0.zip>

copy and unzip target/releases/elasticsearch-analysis-ik-{version}.zip to your-es-root/plugins/ik

重启 elasticsearch

<https://www.elastic.co/>

基本信息查看

Elastic Search 完全是基于 RESTful 接口来操作的，支持操作方法包括 PUT POST GET DELETE 等

查看基本信息

```
curl 'localhost:9200/_cat/health?v'
```

```
epoch timestamp cluster status node.total node.data shards pri relo init unassign
1394735289 14:28:09 elasticsearch green 1 1 0 0 0 0 0
```

<https://www.elastic.co/>

创建索引

```
curl -XPUT 'localhost:9200/htmlcontent?pretty'
```

上面的命令创建了一个叫做 htmlcontent的索引
pretty 是使得打印JSON返回的时候进行格式化

```
curl 'localhost:9200/_cat/indices?v'
```

这个命令用来查看系统已经存在的索引

Python urllib2 的 PUT

```
opener = urllib2.build_opener(urllib2.HTTPHandler)
req = urllib2.Request(base_url + '/html?pretty', data=json.dumps(data))
req.get_method= lambda: 'PUT'
print opener.open(req)
```

需要使用 `build_opener` 来设定 PUT 方法（后面的 DELETE 也一样）

创建 Mapping

为了让搜索引擎知道各个Field的类型，例如 **time**，文本，整数，我们可以对 **index** 创建**mapping**。创建**mapping** 的时候同时指定 **analyzer** 也就是中文分词器

Mapping 支持的数据类型：

- String: string
- Whole number: byte, short, integer, long
- Floating-point: float, double
- Boolean: boolean
- Date: date

创建 Mapping

```
data = {  
  "mappings": {  
    "travel": {  
      "properties": {  
        "title": {  
          "type": "string",  
          "analyzer": "ik_smart"  
        },  
        "content": {  
          "type": "string",  
          "analyzer": "ik_smart"  
        }  
      }  
    }  
  }  
}
```


添加文档

PUT htmlcontent/travel/1

```
{  
  "url" : "http://www.mafengwo.cn/i/1982738.html",  
  "post_date" : "2009-11-15T14:12:12",  
  "title" : "浪漫天堂-塞舌尔"  
  "content" : "抽取的网页正文"  
}
```

这个命令把文档存到了 `htmlcontent` 索引下，`type` 为 `travel` 的类型下，
文档编号为1

添加文档后的信息

```
{
  "_shards" : {
    "total" : 2,
    "failed" : 0,
    "successful" : 2
  },
  "_index" : "htmlcontent",
  "_type" : "travel",
  "_id" : "6a8ca01c-7896-48e9-81cc-9f70661fcb32",
  "_version" : 1,
  "created" : true,
  "result": "created"
}
```

直接获取文档

GET htmlcontent/travel/0

直接使用GET命令，后面指明文档ID，可以直接获取一篇文档

HEAD htmlcontent/travel/0

使用HEAD命令，类似用法，可以检查一篇文档是否存在

查询后删除

POST `htmlcontent/_delete_by_query`

```
curl -XPOST 'localhost:9200/twitter/_delete_by_query?pretty' -H 'Content-Type: application/json' -d'{ "query": { "match": { "message": "some message" } } }'
```

直接基于查询 `_delete_by_query` 命令，在BODY里，指明 `query` 条件来删除

<https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-delete-by-query.html>

更新文档

POST test/type1/1/_update

```
{
  "script" : {
    "inline": "ctx._source.tags.add(params.tag)",
    "lang": "painless",
    "params" :
    {
      "tag" : "blue"
    }
  }
}
```

curl 命令直接更新一个文档

```
curl -XPOST 'localhost:9200/htmlcontent/travel/1/_update?pretty' -H 'Content-Type: application/json' -d'{  "doc" : {    "title" : "new title"  } }
```

Update by Query

POST twitter/_update_by_query

```
{
  "script": {
    "inline": "ctx._source.likes++",
    "lang": "painless"
  },
  "query": {
    "term": {
      "user": "kimchy"
    }
  }
}
```

直接利用 `update` 和 `script` 命令组合，来查询一个文档并修改

查询文档

查询可以直接通过 URI 来进行，即

GET /html/_search?q=content:塞舌尔

也可以通过 QUERY DSL 来查询，支持三种级别：基于Boolean 的 Match、Term 和 Range。Match 是基于针对的查询，Term 是针对单个词的查询，range 是针对数学上的 大于、小于（lt、gt、gte）等

查询文档 – Term

```
data = {  
    "query": {  
        "term":{"content":"塞舌尔"}  
    },  
    "from": 0,  
    "size": 10,  
    "sort":{"_score": "desc"}  
}  
  
req = urllib2.Request(base_url + '/html/_search?pretty', data=json.dumps(data))  
response = urllib2.urlopen(req)  
print response.read()
```


查询文档 - Match

```
data = {  
    "query": {  
        "match": {"content": "塞舌尔 自驾"}  
    },  
    "from": 0,  
    "size": 10,  
    "sort": { "_score": "desc" }  
}
```

```
req = urllib2.Request(base_url + '/html/_search?pretty', data=json.dumps(data))  
response = urllib2.urlopen(req)  
print response.read()
```

查询文档 – Multi Match

```
match_data = {  
  "query": {  
    "multi_match": {  
      "query": "塞舌尔 自驾",  
      "fields": [ "title", "content" ]  
    }  
  },  
  "from": 0,  
  "size": 10,  
  "sort": { "_score": "desc" }  
}
```

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html>

疑问

□ 问题答疑：<http://www.xxwenda.com/>

■ 可邀请老师或者其他回答问题

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop

