云数据管理II 《检索结果分类/聚类》

2015080121 软工52 李在弦
2015013161 软工51 魏佳辰

实验环境

操作系统：Windows
IDE：PyCharm
系统结构：b/s 结构
Web框架：Django 2.1.2
后端编程语言：Python 3.6
数据库：MongoDB

数据库

- ***MongoDB 介绍***

MongoDB 是一个基于分布式文件存储的数据库，它是一个面向文档存储的数据库，操作起来比较简单和容易。如果负载的增加（需要更多的存储空间和更强的处理能力） ，它可以分布在计算机网络中的其他节点上这就是所谓的分片。

- ***分布式存储***

当 MongoDB 存储海量的数据时，一台机器可能不足以存储数据，也可能不足以提供可接受的读写吞吐量。这时，我们就可以通过在多台机器上分割数据，使得数据库系统能存储和处理更多的数据。分片结构端口分布如下：

```
shards:
        { "_id" : "shard0000", "host" : "localhost:27020" }
        { "_id" : "shard0001", "host" : "localhost:27021" }
        { "_id" : "shard0002", "host" : "localhost:27022" }
        { "_id" : "shard0003", "host" : "localhost:27023" }
```

Config server: 27100
Route Process: 40000
启动shard server, config server和route process之后，我们对分片进行配置
首先通过db.runCommand({addShard:”localhost:27020”})等命令添加分片
然后通过db.runCommand({enablesharding:”docdb”})来设置分片存储的数据库
再通过db.runCommand({shardcollection:”docdb.info”,key:{time:1}})来分片info这集合
然后我们通过mongoimport导入处理好的json文件来插入数据，最后数据库自动分片结果如下

```
mongos> db.info.count()
36231
mongos> db.info.getShardDistribution()

Shard shard0000 at localhost:27020
 data : 128.93MiB docs : 12835 chunks : 1
 estimated data per chunk : 128.93MiB
 estimated docs per chunk : 12835

Shard shard0001 at localhost:27021
 data : 0B docs : 0 chunks : 1
 estimated data per chunk : 0B
 estimated docs per chunk : 0

Shard shard0002 at localhost:27022
 data : 135.51MiB docs : 10120 chunks : 1
 estimated data per chunk : 135.51MiB
 estimated docs per chunk : 10120

Shard shard0003 at localhost:27023
 data : 51.01MiB docs : 3155 chunks : 2
 estimated data per chunk : 25.5MiB
 estimated docs per chunk : 1577

Totals
 data : 315.46MiB docs : 26110 chunks : 5
 Shard shard0000 contains 40.87% data, 49.15% docs in cluster, avg obj size on shard : 10KiB
 Shard shard0001 contains 0% data, 0% docs in cluster, avg obj size on shard : NaNGiB
 Shard shard0002 contains 42.95% data, 38.75% docs in cluster, avg obj size on shard : 13KiB
 Shard shard0003 contains 16.17% data, 12.08% docs in cluster, avg obj size on shard : 16KiB
```

算法说明

- ***网络爬虫***

  从 10 万多个单词词库中选取 1700 多个单词，通过 request 库访问百度首页，输入单词，爬取前三个搜索结果的 URL，标题等信息。

  访问每个 URL，通过 BeautifulSoup 库分析 HTML 信息，排除不需要的标签，获取文本信息。

  最后把 URL，标题，文本内容等信息保存到 txt 文件。

  最终爬取到 5222 个文档信息。

- ***分词，提取关键词，获取词干，建立“单词-文档权值矩阵”和倒排索引***

  读取刚刚保存的所有文档的文本信息，提取关键词，进行分词（只对关键词进行分词）。

  进行分词的时候，为了更好地对英文单词进行分词，用 nltk 开源库获取每个关键词的词干，然后用这些词干建立“单词-文档权值矩阵”并保存关键词在文档中出现的次数。

  同时，为了以后实现命中词突出显示的功能，把每个关键词在文档中出现的位置存储到倒排索引。

- ***筛选关键词***

  删除关键词中的不需要的字符

```
del_keyword = "!#$%^&*()_-+=[]{};:/?.>,<、，。‘“；：、】}【{—）（……￥！·~`《》？“‘`"
```

- ***停用词处理***

  设置几十个常见的停用词，然后修改“单词-文档权值矩阵”中每个停用词出现次数。

stop_words.txt

```
the 100
of 100
is 100
and 100
to 100
in 100
that 100
we 100
for 100
an 100
are 100
by 100
be 100
as 100
on 100
with 100
can 100
if 100
from 100
which 100
you 100
it 100
this 100
then 100
at 100
have 100
all 100
not 100
one 100
has 100
or 100
that 100
的 1000
了 100
是 100
在 100
和 100
有 100
他 100
不 100
我 100
人 100
也 100
为 100
```

- ***使用 Tf.idf 扩展布尔模型，计算 mj，tf，dj，nk，idf，Wkj***

  - ***Term frequency (tf):***

$$tf_{ij} = f_{ij} / m_j \qquad \text{when } f_{ij} > 0 \quad m_j = \max(f_{ij})$$

Inverse document frequency (idf):

$$idf_i = \log\left(\frac{n}{n_i}\right) + 1 \qquad n_i > 0$$

- Suppose there are n documents and that the number of documents in which term k_i occurs is n_i.

$$w_{kj} = (f_{kj} / m_j) * (\log(n/n_k) + 1) \qquad \text{when } f_{kj} > 0$$

- ***检索，相关性计算***

  收到查询语句之后对它进行分词，记录每个单词出现次数，计算 mq，Wkq，dq。

$$w_{kq} = (f_{kq} / m_q) * (\log(n/n_k) + 1) \quad \text{when } f_{kq} > 0$$

$$w_{kq} = (0.5 + 0.5 * f_{kq} / m_q) * (\log(n/n_k) + 1) \quad \text{when } f_{kq} = 0$$

  然后计算查询语句和每个文档之间的相关度，

$$\text{sim}(\mathbf{d}_q, \mathbf{d}_j) = \frac{\sum_{k=1}^{n} w_{kq} w_{kj}}{|\mathbf{d}_q| \ |\mathbf{d}_j|}$$

  最后按相关度排序结果。

- **聚类**

  获取所有搜索结果文档的关键词，从“Tf.idf 扩展布尔模型矩阵”获取各个文档和关键词相应的权值。用这些文档编号列表，关键词列表和权值来建立新的“Tf.idf 扩展布尔模型矩阵”。

  Ex）假设下边图是原来的“Tf.idf 扩展布尔模型矩阵”

	ant	bee	cat	dog	eel	fox	gnu	hog
q	1.18	0.59	0.74	1.18	0.74	0.59	0.74	0.74
d_1	1.18	0.59	0	0	0	0	0	0
d_2	0.29	0.29	0	1.18	0	0.29	0	0.37
d_3	0	0	1.48	1.18	1.48	1.18	1.48	0

  其中，如果跟搜索语句有相关的文档是 d2 和 d3，它们两个的关键词并集是“ant，bee，cat”，那么新的“Tf.idf 扩展布尔模型矩阵”是

	ant	bee	cat
d_2	0.29	0.29	0
d_3	0	0	1.48

  用这些权值来进行动态聚类。

  聚类部分用了动态聚类方法中的重心法。首先计算全部聚类文档权值的重心，建立新凝聚点的最小临界距离 d，依次逐个输入全部文档权值，如果输入文档与已有凝聚点的距离大于 d，则将它作为新的凝聚点；否则不作为凝聚点；

最后，把每个类的关键词权值分别加，然后用权值最高的关键词来代表每个类

Ex) ant = 0.29 + 0 = 0.29

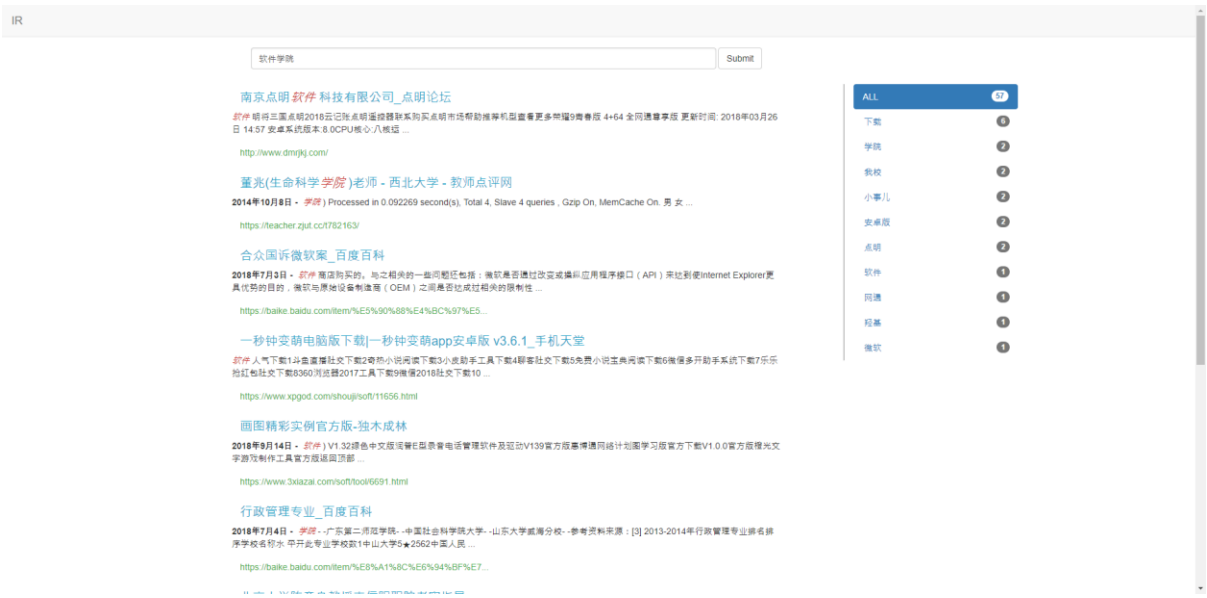
bee = 0.29 + 0 = 0.29

cat = 0 + 1.48 = 1.48

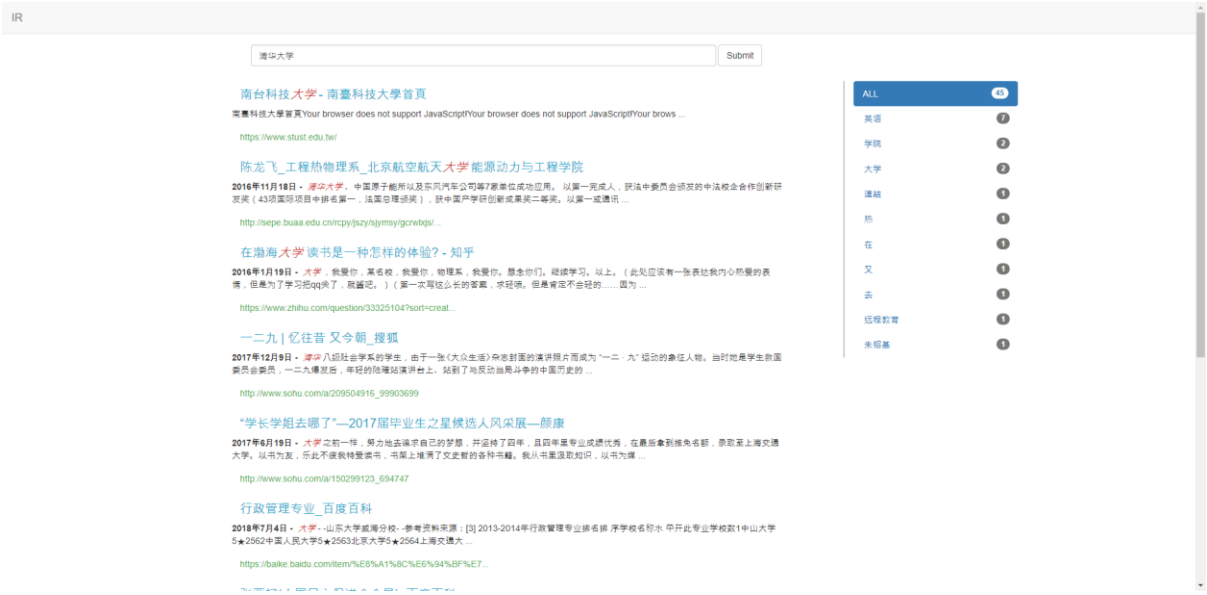
所以权值最高的关键词是“cat”，所以用它代表类名

可视化

《输入“亚运会”的搜索结果》



《输入“软件学院”的搜索结果》



《输入“清华大学”的搜索结果》

《输入“python”的搜索结果》

使用说明

（先安装好 nltk库，pymongo库，jieba库，requests库，bs4库，hashlib库，MongoDB， django ）

1. 执行src/preprocess目录下的pachong.py文件（爬虫）

   修改pachong.py文件的188行就能调整爬虫文档数。

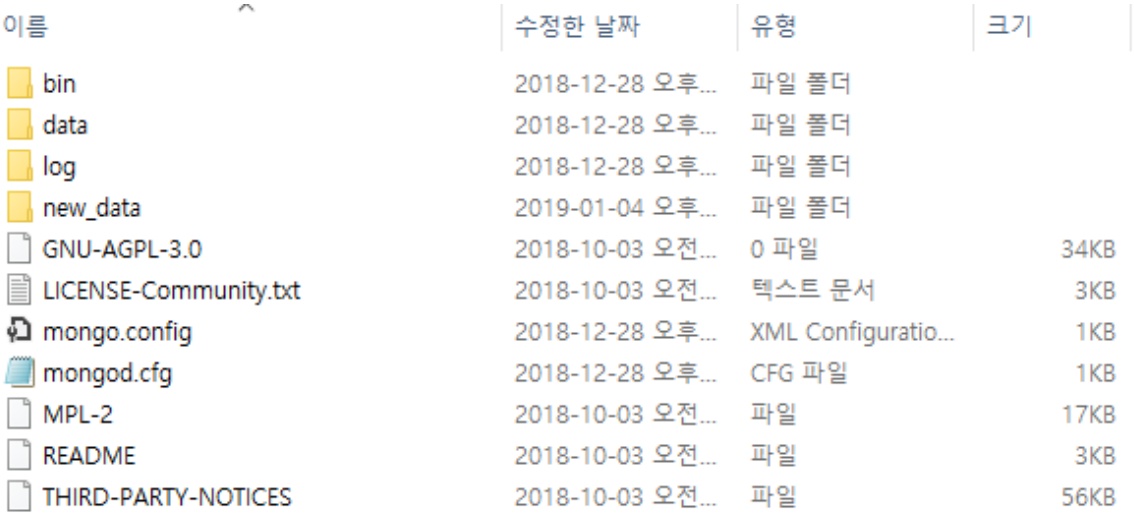
   ```
   188        if countNum % 65 == 1:
   ```

   （修改65）

   （因为爬虫需要两三个小时，为了让助教老师更方便运行我们的程序，我把5222个文档打包之后放在src/preprocess/pachong目录下。）

2. 启动MongoDB

   打开cmd，进入MongoDB目录，

   创建新的文件夹（比如，new_data）

   进入MongoDB/bin目录

   然后运行 mongod --dbpath (new_data文件夹的路径)

   ```
   C:₩MongoDB₩bin>mongod --dbpath c:₩MongoDB₩new_data
   ```

   就可以启动MongoDB。

3. 执行src/preprocess目录下的fenci.py文件

   （注意！运行程序的时候必须从pachong文件夹删除打包文件，pachong.zip）

   运行fenci.py文件做

   “分词”，“ 提取关键词”，“ 获取词干”，

   “ 建立单词-文档权值矩阵和倒排索引”，“筛选关键词”，“ 停用词处理”，

   “ 使用 Tf.idf 扩展布尔模型，计算 mj，tf，dj，nk，idf，Wkj”

   这些工作。

做完这些工作之后会把数据保存到MongoDB。

4. 打开django服务器
   打开cmd，进入src目录，执行 python manage.py runserver 0:8000 --noreload
   每次打开服务器的时候需要从数据库获取 tf.idf扩展布尔模型矩阵，所以可能需要2~3分钟的时间。
5. 进入网站
   打开浏览器，访问 127.0.0.1:8000