



南开大学
Nankai University

南 开 大 学

计 算 机 学 院

信息检索系统实验报告

hw4_anime_search_engine

姓名：王泳鑫

学号：1911479

年级：2019级

专业：计算机科学与技术

指导教师：温延龙

2021 年 12 月 14 日

摘要

关键字：倒排索引，index，elasticsearch,enron

目录

一、 实验描述	1
二、 实验设计与代码实现	1
(一) 网页抓取	1
(二) 构建索引	2
(三) 链接分析(未实现)	3
(四) 查询服务	4
1. 全文查询	4
2. 站内查询	5
3. 时间区间查询	5
4. 短语查询	5
5. 通配查询	6
6. 网页快照	6
7. 查询历史	6
(五) 个性化查询	7
1. 查询历史	7
(六) 用户交互UI	7
1. 登录与注册	7
2. 功能UI	9
三、 实验演示与结果分析	10

一、 实验描述

本次作业的要求是针对南开校内资源构建一个Web搜索引擎，为用户提供南开信息的查询服务和个性化推荐。我爬取了南开12club动漫站的资源来构建一个基于python和elasticsearch简易的搜索引擎，同时使用了kivy来完成用户的交互界面。

二、 实验设计与代码实现

（一） 网页抓取

这里我对南开12club进行了一个抓取，但是由于12club网站过于扁平化，没有那么多复杂的链接关系，因此最后爬取了两千多个文档，爬取的过程就是通过request包中get方法获取url的html文件，通过浏览器的检查功能，我们很快就能找到我们需要的有效信息，然后通过正则匹配来获取这些有效信息，代码如下：

```
1 def get_one_page(url,headers):
2     try:
3         response = requests.get(url,headers = headers)
4         response.encoding = 'utf-8'
5         if response.status_code == 200:
6             return response.text
7         return None
8     except RequestException:
9         return None
10
11 def parse_one_page(html):
12     p = re.compile('<title>.*?</title>',re.S)
13     Item = re.findall(p,html)
14
15     p=re.compile("</td><td class='val' colspan='3'>.*?</td>",re.S)
16     Item2=re.findall(p,html)
17
18
19     pattern = re.compile("</td><td class='val'>.*?</td>",re.S)
20     items = re.findall(pattern, html)
21     if(len(items)==7):
22         yield {
23             'name':str(Item).replace('<title>','').replace('</title>',''),
24             'othername':str(Item2).replace("</td><td class='val' colspan='3'>","").replace('</td>',''),
25             'subtitleMade': str(items[0]).replace("</td><td class='val'>","").replace('</td>',''),
26
27             'type': str(items[1]).replace("</td><td class='val'>","").replace('</td>',''),
28             'updatetime': str(items[2]).replace("</td><td class='val'>","").replace('</td>',''),
29
30             'commentsN': str(items[3]).replace("</td><td class='val'>","").replace('</td>',''),
31             'episodes': str(items[4]).replace("</td><td class='val'>","").replace('</td>',''),
```

```

32         'downloadN':str(items[5]).replace("</td><td class='val'>","").replace(
33             '</td>',''),
34         'state':str(items[6]).replace("</td><td class='val'>","").replace('</td>',''),
35         'downloadNweek':str(items[7]).replace("</td><td class='val'>","").
36             replace('</td>','')
37     }
38 else:
39     return

```

(二) 构建索引

这里我通过es来构建索引，与hw3类似，首先还是在es中创建一个索引，代码如下：

```

1 def create_index():
2     if es.indices.exists(INDEX_NAME):
3         es.indices.delete(INDEX_NAME)
4         print('删除存在的索引 \'{ }\', 并创建一个新的索引'.format(INDEX_NAME))
5     result = es.indices.create(index=INDEX_NAME, ignore=400)
6     print(result)
7     put_map()
8     return

```

然后确定索引的字段以及索引字段的类型，还是用到了es的put_mapping方法，代码如下：

```

1 def put_map():
2     es.indices.put_mapping(
3         body=
4         {
5             "dynamic": "strict",
6             #"_source": {"enabled": "false"},
7             "properties":
8             {
9                 "url":{"type": "keyword"}
10                ,"name": {"type": "keyword"}
11                ,"othername": {"type": "keyword"}
12                ,"subtitleMade": {"type": "keyword"}
13                ,"type": {"type": "keyword"}
14                ,"updateTime": {"type": "date","format": "yyyy-MM-dd HH:mm:ss"}
15                ,"commentsN": {"type": "keyword"}
16                ,"episodes": {"type": "keyword"}
17                ,"downloadN": {"type": "keyword"}
18                ,"state": {"type": "keyword"}
19                ,"downloadNweek": {"type": "keyword"}
20            }
21        }
22        ,index=INDEX_NAME
23        ,doc_type='12club'
24        ,include_type_name=True
25    )

```

最后就是插入索引的过程，我们需要把爬取的有效信息插入es建立索引之中，代码如下：

```

1 def index_to_es(url,html):
2     print("start!!")
3     p = re.compile('<title>.*</title>', re.S)

```

```

4     Item = re.findall(p, html)
5
6     p = re.compile("</td><td class='val' colspan='3'>.*?</td>", re.S)
7     Item2 = re.findall(p, html)
8
9     pattern = re.compile("</td><td class='val'>.*?</td>", re.S)
10    items = re.findall(pattern, html)
11    print(items)
12    if (len(items) == 8):
13        josndoc = {
14            'url':url,
15            'name': str(Item[0]).replace('<title>', '').replace('</title>', '').
                replace(' - 动画 - 十二社
                区', '').replace(' - 漫画 - 十二社
                区', '').replace(' - 音乐 - 十二社
                区', '').replace(' - 游戏 - 十二社
                区', '').replace(' - 轻小说 - 十二
                社区', '').replace(' - 视频 - 十二
                社区', '').replace('完[]', ''),
16            'othername': str(Item2[0]).replace("</td><td class='val' colspan='3'>", ''
                ).replace('</td>', ''),
17            'subtitleMade': str(items[0]).replace("</td><td class='val'>", '').replace
                ('</td>', ''),
18
19            'type': str(items[1]).replace("</td><td class='val'>", '').replace('</td>'
                , ''),
20            'updatetime': str(items[2]).replace("</td><td class='val'>", '').replace('
                </td>', ''),
21
22            'commentsN': str(items[3]).replace("</td><td class='val'>", '').replace('
                </td>', ''),
23            'episodes': str(items[4]).replace("</td><td class='val'>", '').replace('</
                td>', ''),
24            'downloadN': str(items[5]).replace("</td><td class='val'>", '').replace('
                </td>', ''),
25            'state': str(items[6]).replace("</td><td class='val'>", '').replace('</td>
                ', ''),
26            'downloadNweek': str(items[7]).replace("</td><td class='val'>", '').
                replace('</td>', '')
27        }
28    print(josndoc)
29    try:
30        es.index(index=INDEX_NAME, doc_type="12club", body=josndoc)
31        print("Done indexing the doc")
32    except Exception as ex:
33        traceback.print_exc()
34        print("Failed to index the document {}".format(jsonMapDoc))
35    else:
36        pass

```

(三) 链接分析(未实现)

由于我爬取的这个网站过于扁平，对于大部分网站来说，以至于在对某一个网站进行链接分析的过程中，从该网页链接出去的网页就只有该网页本身，导致每个网页的pagerank区别不大，没有参考的意义，因此我在这里只简单阐述PageRank的原理。

我们通过有向图可以表示链接之间的跳转关系，边上的权值为随机跳转的概率，最后，一

个网页被指向的链接越多，直观上就表示跳转这个网页的概率就越高，因此它的重要程度就越高，PageRank就越高。

```

1  # 使用计算节点的networkXpagerank
2  import networkx as nx
3  import matplotlib.pyplot as plt
4
5  # 创建有向图
6  G = nx.DiGraph()
7  # 设置向有图的边集合
8  edges = [("A", "B"), ("A", "C"), ("A", "D"), ("B", "A"), ("B", "D"), ("C", "A"), ("D",
9  "B"), ("D", "C")]
10
11 # 在有向图中添加边集合G
12 for edge in edges:
13     G.add_edge(edge[0], edge[1])
14
15 # 有向图可视化
16 # layout = nx.spring_layout(G)
17 # layout = nx.circular_layout(G)
18 layout = nx.shell_layout(G)
19 # nx.draw(G, pos=layout, with_labels=True, hold=False)
20 nx.draw(G, pos=layout, with_labels=True)
21 plt.show()
22
23 # 计算简化模型的值PR
24 pr = nx.pagerank(G, alpha=1)
25 print("简化模型的值: PR", pr)
26
27 # 计算随机模型的值PR
28 pr = nx.pagerank(G, alpha=0.85)
29 print("随机模型的值: PR", pr)

```

以上代码来自CSDN。

(四) 查询服务

1. 全文查询

全文查询即我检索的信息需要在除更新时间外的所有字段来检索，代码如下：

```

1  def simpleSearch(input):
2      query = {
3          "query": {
4              "multi_match": {
5                  "query": input,
6                  "fields": ["name^3", "othername^3", "subtitleMade", "type", "commentsN", "
7                  episodes", "downloadN", "
8                  state", "downloadNweek"]
9              }
10         },
11         "from": 0,
12         "size": 10,
13         "sort": [],
14         "aggs": {}
15     }
16     res = es.search(index=INDEX_NAME, body=query)

```

```
15     return res
```

2. 站内查询

站内查询通常是用户输入一个url，然后再键入一个关键词，在这个网站内进行检索，代码如下：

```
1 def OnsiteSearch(url, input):
2     query = {"query": {"bool": {"must": [{"match": {"type": url[-1]}, {"match": {"name":
                                     input}}], "must_not": [], "should": []}}, "
                                     from": 0, "size": 10, "sort": [], "aggs": {}}}
3     res = es.search(index=INDEX_NAME, body=query)
4     return res
```

我们根据url进行分析，可以发现站内查询的url最后一位对应这索引的type字段，因此我们match该字段再进行全文检索即可。

3. 时间区间查询

时间区间查询，每个动漫或漫画等都有一个更新时间的字段，同时这次作业中建立更新时间字段的过程没有遇到太多困难，再putmap的时候把字段的格式固定，就可以了，代码如下：

```
1 def dateSearch(time1, time2):
2     query = {"query": {"bool": {"must": [{"range": {"update_time": {"gt": time1, "lt": time2}}}
                                     ], "must_not": [], "should": []}}, "from": 0,
                                     "size": 10, "sort": [], "aggs": {}}}
3     res = es.search(index=INDEX_NAME, body=query)
4     return res
```

4. 短语查询

对于短语查询，es内置的分析器对中文来说不太友好，把关键词一个一个分开来检索，这样对检索系统来说，会检索到很多很多不相关的文档，因此在这里我下载了一个ik中文分词器，这样的短语划分对于中文来说更有效，代码如下：

```
1 def phraseSearch(input):
2     query = {
3         "query": {
4             "multi_match": {
5                 "query": input,
6                 "fields": ["name", "othername"],
7                 "type": "phrase"
8                 "analyzer": "ik_smart"
9             }
10        },
11        "from": 0,
12        "size": 10,
13        "sort": [],
14        "aggs": {}
15    }
16    res = es.search(index=INDEX_NAME, body=query)
17    return res
```

5. 通配查询

es的通配查询就是通配符查询，支持*(匹配任意字符)和?(匹配单个字符)，代码如下：

```
1 def wildcardSearch(input):
2     s=input+'*'
3     query = {
4         "query": {
5             "bool": {
6                 "should": [
7                     {
8                         "wildcard": {
9                             "name": s
10                        }
11                    },
12                    {
13                        "wildcard": {
14                            "othername": s
15                        }
16                    }
17                ]
18            }
19        },
20        "from": 0,
21        "size": 10,
22        "sort": [],
23        "aggs": {}
24    }
25    res = es.search(index=INDEX_NAME, body=query)
26    return res
```

6. 网页快照

对于网页快照，在这里我把检索到url，利用request包get方法得到html文件存入在本地文件夹，代码如下：

```
1 def getHtml(url):
2     html = urllib.request.urlopen(url).read()
3     return html
4
5
6 def saveHtml(file_name, file_content):
7     file_name="./htmlStore/"+file_name.replace('/', '_')
8     with open(file_name + ".html", "wb") as f:
9         f.write(file_content)
10
11
12 def storeHtml(url):
13     saveHtml(url, getHtml(url))
```

7. 查询历史

新建了一个database.py文件来操作文件的读写，同时还有两个文件，一个user.txt来存储用户信息，另一个history.txt来存储某个用户的检索历史，代码如下：


```

1  def show(self):
2      if len(db.load_history(MainWindow.current)) != 0:
3          s = ''
4          for a in db.load_history(MainWindow.current):
5              s = s + str(a) + '\n'
6          #histt = s
7          self.ids.his_label.text = s
8          return s
9      else:
10         s = "search history is empty!!"
11         self.ids.his_label.text = s
12         return "search history is empty!!"

```

（五） 个性化查询

个性化查询这里，只实现了用户的查询历史，没有更进一步进行开发。

1. 查询历史

用户的查询历史都存在history.txt之中，通过database类来控制，代码如下：

```

1  def save_history(self, ee, history):
2      with open(self.filename2, "a", encoding="utf-8") as f:
3          f.write(ee + ";" + history + ";" + self.get_date() + "\n")
4
5
6  def load_history(self, email):
7      self.file = open(self.filename2, "r", encoding="utf-8")
8      historyl = []
9
10     for line in self.file:
11         e, history, searchtime = line.strip().split(";")
12         if (e == email):
13             historyl.append((history, searchtime))
14
15     return historyl

```

（六） 用户交互UI

本次实验的用户UI设计是基于python和kivy来实现，基本上实现了登录登出和注册，还有功能的设计和查询历史的展示。

1. 登录与注册

kivy开发与java的GUI开发很像，每个窗口基本上都要新建一个class，注册登录和主界面代码如下：

```

1  class CreateAccountWindow(Screen):
2      namee = ObjectProperty(None)
3      email = ObjectProperty(None)
4      password = ObjectProperty(None)
5

```

```
6     def submit(self):
7         if self.namee.text != "" and self.email.text != "" and self.email.text.count("@") == 1 and self.email.text.count(".") > 0:
8             if self.password != "":
9                 db.add_user(self.email.text, self.password.text, self.namee.text)
10
11                 self.reset()
12
13                 sm.current = "login"
14             else:
15                 invalidForm()
16         else:
17             invalidForm()
18
19     def login(self):
20         self.reset()
21         sm.current = "login"
22
23     def reset(self):
24         self.email.text = ""
25         self.password.text = ""
26         self.namee.text = ""
27
28
29 class LoginWindow(Screen):
30     email = ObjectProperty(None)
31     password = ObjectProperty(None)
32
33     def loginBtn(self):
34         if db.validate(self.email.text, self.password.text):
35             MainWindow.current = self.email.text
36             self.reset()
37             sm.current = "main"
38         else:
39             invalidLogin()
40
41     def createBtn(self):
42         self.reset()
43         sm.current = "create"
44
45     def reset(self):
46         self.email.text = ""
47         self.password.text = ""
48
49
50 class MainWindow(Screen):
51     n = ObjectProperty(None)
52     created = ObjectProperty(None)
53     email = ObjectProperty(None)
54     current = ""
55
56     def logOut(self):
57         sm.current = "login"
58
```

```

59     def on_enter(self, *args):
60         password, name, created = db.get_user(self.current)
61         self.n.text = "Account Name: " + name
62         self.email.text = "Email: " + self.current
63         self.created.text = "Created On: " + created

```

2. 功能UI

功能UI就是我实现的检索系统的UI，代码如下：

```

1  class SearchWindow(Screen):
2      pass
3
4
5
6  class DocSearchWindow(Screen):
7      a=ObjectProperty(None)
8
9      def search(self):
10         if(MainWindow.current!='' and self.a.text != ''):
11             db.save_history(MainWindow.current,self.a.text)
12             searchRes = Search.printres(Search.simpleSearch(self.a.text))
13
14         def reset(self):
15             pass
16
17
18  class OnsiteSearchWindow(Screen):
19      url1=ObjectProperty(None)
20      a=ObjectProperty(None)
21
22      def search(self):
23         if (MainWindow.current != '' and self.a.text != ''):
24             db.save_history(MainWindow.current, self.a.text)
25             searchRes = Search.printres(Search.OnsiteSearch(self.url1.text,self.a.text
26                                                         ))
27
28         def reset(self):
29             pass
30
31
32  class DateSearchWindow(Screen):
33      time1 = ObjectProperty(None)
34      time2 = ObjectProperty(None)
35
36      def search(self):
37         if (MainWindow.current != '' and self.time1.text != '' and self.time2.text !=
38                                                         ''):
39             db.save_history(MainWindow.current, self.time1.text+self.time2.text)
40             searchRes = Search.printres(Search.dateSearch(self.time1.text, self.time2.
41                                                         text))
42         print(searchRes)

```

```

43     def reset(self):
44         pass
45
46 class PhraseSearchWindow(Screen):
47     a = ObjectProperty(None)
48
49     def search(self):
50         if (MainWindow.current != '' and self.a.text != ''):
51             db.save_history(MainWindow.current, self.a.text)
52             searchRes = Search.printres(Search.phraseSearch(self.a.text))
53
54     def reset(self):
55         pass
56
57 class WildCardSearchWindow(Screen):
58     a = ObjectProperty(None)
59
60     def search(self):
61         if (MainWindow.current != '' and self.a.text != ''):
62             db.save_history(MainWindow.current, self.a.text)
63             searchRes = Search.printres(Search.wildcardSearch(self.a.text))
64         pass
65
66     def reset(self):
67         pass
68
69
70 class ResWindow(Screen):
71
72     def show(self):
73         print(searchRes)
74         self.ids.res_label.text = searchRes
75         return "res"
76
77 class SearchHistoryWindow(Screen):
78
79     def show(self):
80         if (len(db.load_history(MainWindow.current)) != 0):
81             s=''
82             for a in db.load_history(MainWindow.current):
83                 s=s+str(a)+'\n'
84             #histt=s
85             self.ids.his_label.text = s
86             return s
87         else:
88             s="search history is empty!!"
89             self.ids.his_label.text = s
90             return "search history is empty!!"

```

三、 实验演示与结果分析

登录界面如图5所示

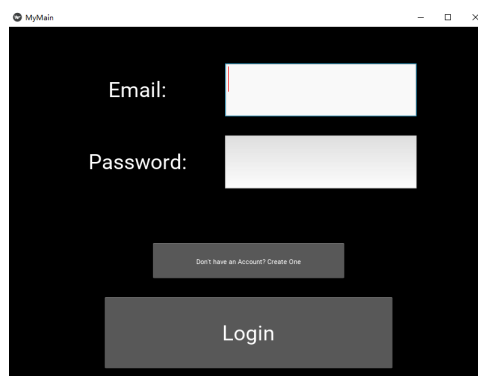
A screenshot of a web application window titled 'MyMain'. The background is black. It features a login form with two white input fields: 'Email:' and 'Password:'. Below the password field is a link that says 'Don't have an Account? Create One'. At the bottom is a large grey button labeled 'Login'.

图 1: 登录界面

注册界面如图5所示

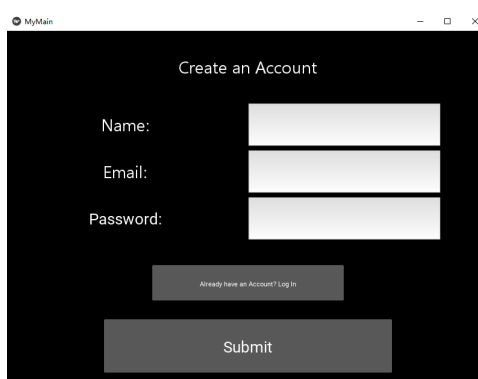
A screenshot of a web application window titled 'MyMain'. The background is black. It features a registration form titled 'Create an Account' with three white input fields: 'Name:', 'Email:', and 'Password:'. Below the password field is a link that says 'Already have an Account? Log In'. At the bottom is a large grey button labeled 'Submit'.

图 2: 注册界面

登陆状态如图5所示

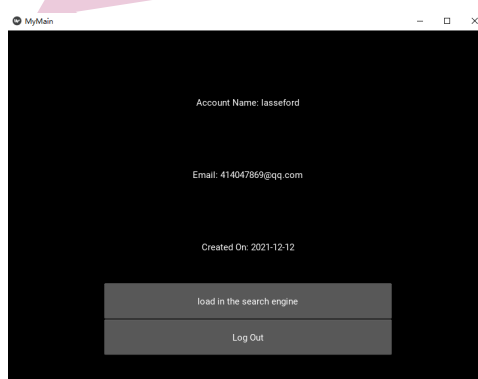
A screenshot of a web application window titled 'MyMain'. The background is black. It displays user information: 'Account Name: lasseford', 'Email: 414047869@qq.com', and 'Created On: 2021-12-12'. At the bottom are two grey buttons: 'load in the search engine' and 'Log Out'.

图 3: 登陆状态

查询功能如图5所示

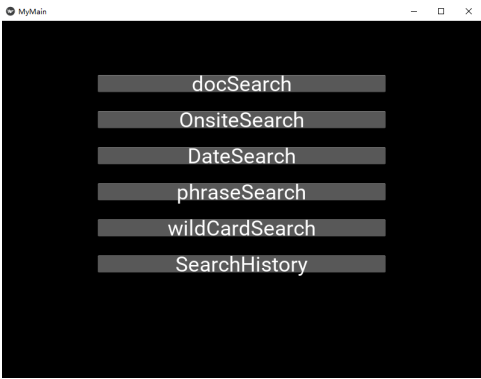


图 4: 查询功能

查询历史如图5所示



图 5: 查询历史