

**本科毕业设计（论文）**

题 目 校园网搜索引擎的设计与实现

学 院 计算机科学与工程

年 级 2014 专 业 软件工程（嵌入式）

班 级 Z094142 学 号 Z09414211

学生姓名 徐德安

校内导师 沈健 职 称 高级实验师

校外导师 孙忠建 职 称 高级教员

论文提交日期

常熟理工学院本科毕业设计(论文)诚信承诺书

本人郑重声明： 所呈交的本科毕业设计(论文)，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

|  |  |  |  |
| --- | --- | --- | --- |
| 本人签名： |  | 日期： | 2018年4月5日 |

常熟理工学院本科毕业设计(论文)使用授权说明

本人完全了解常熟理工学院有关收集、保留和使用毕业设计(论文)的规定，即：本科生在校期间进行毕业设计(论文)工作的知识产权单位属常熟理工学院。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许毕业设计(论文)被查阅和借阅；学校可以将毕业设计(论文)的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编毕业设计（论文），并且本人电子文档和纸质论文的内容相一致。

保密的毕业设计(论文)在解密后遵守此规定。



|  |  |  |  |
| --- | --- | --- | --- |
| 本人签名： |  | 日期： | 2018年4月5日 |
| 导师签名： |  | 日期： |  |

**校园网站搜索引擎的设计与实现**

**摘要**

随着智慧校园的建设的推进，各学院、部门建立了大量的网站，网站内的资源越来多，而且很多网站仅限校内访问，随之而来的问题就是，无法快速找到校内网站的内容。

针对上述问题，本论文以“常熟理工学院校园新闻网”和二级学院网站“计算机科学与工程学院网站”为对象，在充分分析校园网结构基础之上，结合ElasticSearch搜索引擎和Django框架设计并实现一款基于Python技术、应用于校园网站的站内搜索引擎。

首先，使用Scrapy爬虫框架从校园网中爬取数据，并建立索引库。用户可以在交互页面输入想要查询的关键字。搜索引擎接收关键字后检索索引库，并返回用户所需要的信息。最后，对原型系统进行测试，并对搜索结果的正确性和效率进行分析来充分说明本系统的可用性。

通过测试可以发现，该搜索引擎能够有效工作，提供相对准确搜索结果。相比其他搜索引擎和现有的搜索技术，大大提高了搜索效率和准确度。

**关键词：**搜索引擎 Scrapy 校园网 ElasticSearch

Design and implementation of campus website search engine

Abstract

With the development of the smart campus, many colleges and departments have established a large number of websites, more and more resources in the website, and many websites are limited to campus access. The following problem is that it is impossible to quickly find the content of the campus website.

In view of the above problems, this thesis takes the campus website of Changshu Institute Of Technology as the object. On the basis of fully analyzing the structure of the campus network, it combines the ElasticSearch search engine and the Django framework to design and implement a site search engine based on Python technology and applied to campus web site.

First, we use Scrapy crawler framework to crawl data from campus network and build index library. Users can input keywords that are to be querying on the interaction page. The search engine keyword after receiving inspection cable lead base, and returns the information needed by the user. Finally, the prototype system is tested, and the correctness and efficiency of search results are analyzed to fully illustrate the usability of the system.

Through testing, it can be found that the search engine can work effectively and provide relatively accurate search results. Compared with other search engines and existing search technologies, the efficiency and accuracy of search are greatly improved.

**Key Words:** Search Engines; Scrapy; Campus network;ElaticSearch

目 录

[1 绪论 1](#_Toc510888984)

[1.1 课题背景 1](#_Toc510888985)

[1.2 研究意义 1](#_Toc510888986)

[1.3 本文主要工作 1](#_Toc510888987)

[2 搜索引擎的工作原理及相关知识介绍 2](#_Toc510888988)

[2.1 搜索引擎简介 2](#_Toc510888989)

[2.2 搜索引擎工作流程 2](#_Toc510888990)

[2.3 网络爬虫 3](#_Toc510888991)

[2.3.1宽度优先策略 4](#_Toc510888992)

[2.3.2 深度优先策略 4](#_Toc510888993)

[2.4搜索引擎索引 5](#_Toc510888994)

[2.4.1索引 5](#_Toc510888995)

[2.4.2 匹配 7](#_Toc510888996)

[2.4.3排名 7](#_Toc510888997)

[3 Scrapy在校园网站搜索中的应用 10](#_Toc510888998)

[3.1 Scrapy介绍与使用 10](#_Toc510888999)

[3.1.1 Scrapy简介 10](#_Toc510889000)

[3.1.2 Scrapy的使用 11](#_Toc510889001)

[3.2 Scrapy中常见的问题 12](#_Toc510889002)

[3.2.1 URL去重策略 12](#_Toc510889003)

[3.2.3 关于反爬虫以及应对反爬虫的策略 13](#_Toc510889004)

[4 校园搜索引擎系统的设计与实现 15](#_Toc510889005)

[4.1 系统总体架构设计 15](#_Toc510889006)

[4.2 “常熟理工学院新闻网”内容抓取模块 15](#_Toc510889007)

[4.2.1 爬取流程分析 16](#_Toc510889008)

[4.2.2 新闻页面分析 17](#_Toc510889009)

[4.3 “计算机科学与工程学院”内容抓取模块 21](#_Toc510889010)

[4.4 索引与检索模块 23](#_Toc510889011)

[4.4.1 ElasticSearch介绍 23](#_Toc510889012)

[4.4.2 ElasticSearch安装与配置 24](#_Toc510889013)

[4.4.3保存数据到ElasticSearch中 25](#_Toc510889014)

[4.5 前端展示模块 29](#_Toc510889015)

[4.5.1 Django介绍 29](#_Toc510889016)

[4.5.2 Django的使用 30](#_Toc510889017)

[4.5.3 搜索引擎的开发流程 31](#_Toc510889018)

[4.5.4 搜索引擎用户界面与结果展示 35](#_Toc510889019)

[5 系统测试 37](#_Toc510889020)

[5.1 测试环境 37](#_Toc510889021)

[5.2 测试结果分析 37](#_Toc510889022)

[5.3 测试总结 38](#_Toc510889023)

[结束语 39](#_Toc510889024)

[参考文献与附录 40](#_Toc510889025)

[致谢 41](#_Toc510889026)

# 1 绪论

## 课题背景

校园网是老师同学获取学校信息最便捷的来源。你可以在校园网浏览拟想要了解的内容。目前随着互联网的告诉发展，校园的信息化建设越来越完善。几乎每个学校目前都拥有自己的校园网站。校园网站中又分设了下级学院、机构。信息的搜索越来越复杂，越来越困难。又由于某些网站只能通过内网访问，因此搜索引擎很难抓取，通过百度等搜索引擎几乎获取不到自己想要的信息。

因此，开发一款面向校内和全校师生的校园网搜索引擎具有极强的现实意义和实用价值。

## 研究意义

本课题研究的目的主要是针对上述问题，利用基于Python的一些开源框架、技术等，实现一个校园网站内部搜索引擎。

本课题研究的意义在于：

1. 根据常熟理工校园网现有的搜索系统来看，目前的搜索系统只能够根据新闻标题进行检索，这样的搜索结果命中率不高很满足用户的实际需求。
2. 本系统虽然只时初级原型系统，但与原来的搜索系统相比，大大提高了搜索效率，优化了用户体验。
3. 通过进行系统的设计与开发，对Python编程语言也起到了一定的推广作用。可以作为想要学习该技术的同学的学习样本，有一定的参考价值。

## 本文主要工作

根据拟提出的问题，本文所做的工作如下：

1. 根据当前校园网搜索系统来看，采用的是一般的数据库模糊匹配的方式，通过Java语言实现，效率低，查询局限性大。因此采用Python语言结合一些开源框架打造全新的搜索引擎。
2. 把Scrapy应用到原型系统中，选用合适的URL去重策略，并根据实际的反爬虫策略选用合适的解决方案。
3. 采用IK分词器，提高中文分词能力，提高搜索的命中率。
4. 使用Django开发本系统的用户搜索界面，提供良好的人机交互功能。

# 2 搜索引擎的工作原理及相关知识介绍

## 2.1 搜索引擎简介

搜索引擎（Search Engine）是指根据一定的策略、运用特定的计算机程序从互联网上搜集信息。在对信息进行一些列的组织和处理后，为用户提供检索服务，最后将用户检索相关的信息反馈给用户的系统。搜索引擎分类主要包括全文索引、目录索引、元搜索引擎、垂直搜索引擎、集合式搜索引擎、门户搜索引擎以及免费链接列表等[1]。

## 2.2 搜索引擎工作流程

总的来说，搜索引擎的工作原理分为三步：网页收集、预处理、查询服务[4]。

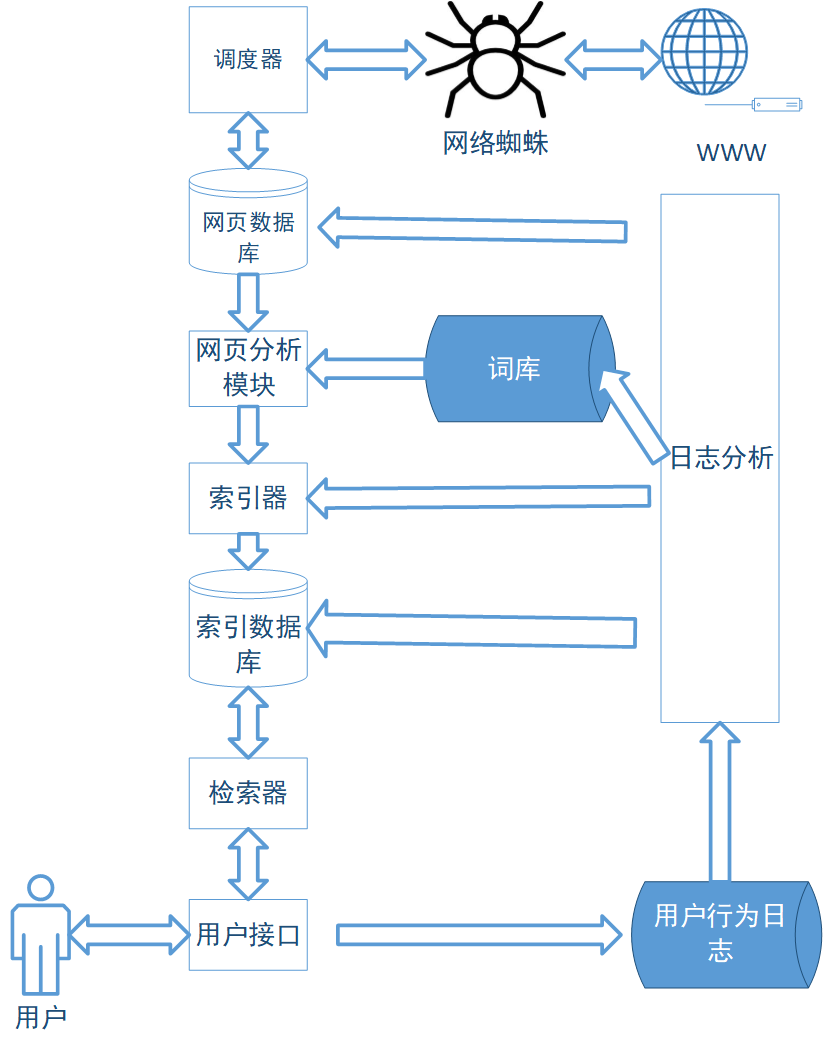


图2-1 搜索引擎工作流程

1. 网页收集与预处理的流程大致如下：

第1步：搜索引擎派出爬虫（spider蜘蛛），按照一定的爬取策略抓取网页到服务器；

第2步：对抓取的网页进行一定的处理，比如链接抽离、内容提取、消除噪音等；

第3步：对网页文本内容进行中文分词、去除无意义的词（如语气词，啊，哦等）；

第4步：对网页分词后进行URL去重操作，对剩余的网页进行倒排索引，为用户提供检索服务。

1. 当用户想要查询信息时，搜索引擎的工作流程大致如下：

第1步：对用户想要查询的关键字进行分词处理，再根据用户的地理位置信息和历史搜索特征进行用户需求分析，然后个性化的展示用户搜索结果。

第2步：查询缓存中是否有该关键词的查询历史和结果，如查有则根据具体用户个性化显示结果。

第3步：如果用户查询的关键词不存在缓存中，则在索引库中的网页进行调取相关排名并呈现给用户，然后将与关键词对应的搜索结果加入到缓存中，以便下一位用户时直接使用。

第4步：网页不仅需要对索引库中的网页进行相关性和重要性的排序，还需要用户体验数据，即用户搜索后的点击以及其他一系列的搜索操作，因为这样也影响到网页的排名。

## 2.3 网络爬虫

我们日常访问的网站大多呈现多层目录结构，也成为树形结构。即网站有一个主页入口，其下又分别设置多个模块，每个模块下又含有多重内容。根据分析，“常熟理工学院”校园网网站结构如下：

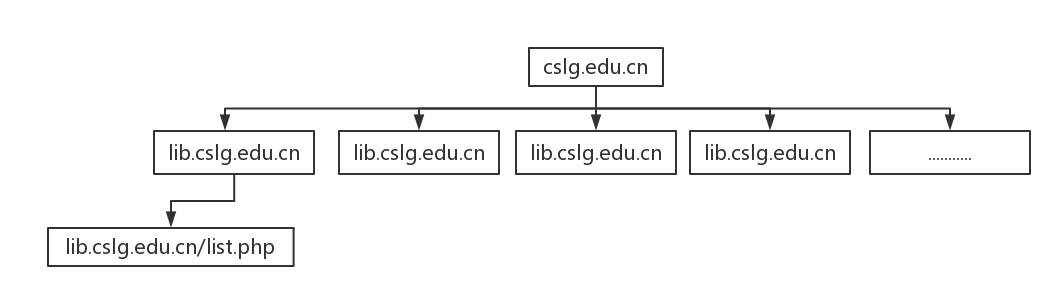


图2-2 “常熟理工”校园网站结构图

网络爬虫为了能够爬取网站的全部资源，通常会选择两种方式：深度优先策略和宽度优先策略[5]。

### 2.3.1宽度优先策略

宽度优先策略，又叫广度优先搜索，通过队列实现，从上往下对每一层的解点依次访问在每一层中，从左往右访问叶子结点，访问完本层后进入下一层继续访问，直到没有结点则停止访问。

**伪代码如下：**

|  |
| --- |
| 初始化队列Q；visited[n]=0；  访问顶点v；visited[v]=1；顶点v入队列Q；  while（队列Q非空）  v=队列Q的对头元素出队；  w=顶点v的第一个邻接点；  while（w存在）  如果w未访问，则访问顶点w；  visited[w]=1；  顶点w入队列Q；  w=顶点v的下一个邻接点。 |

### 2.3.2 深度优先策略

深度优先策略通过递归算法来实现。首先对每一个可能的分支路径进行遍历直到没有子节点为止，并且每个叶子结点只能访问一次。与宽度便利不同，二叉树的深度优先遍历比较特殊，可以使用多种方法，可以细分为先序遍历、中序遍历以及后序遍历

**伪代码如下：**

|  |
| --- |
| 访问顶点v; visited[v]=1;  w=顶点v的第一个邻接点；  while (w存在)  if (w未被访问) 从顶点w出发递归执行该算法;  w=顶点v的下一个邻接点; |

## 2.4搜索引擎索引

搜索有两个重要阶段：匹配、排名。搜索引擎将着两个阶段合成了一个完整的流程[6]。但是，这两个阶段在概念上是独立的，接下来将分别介绍什么是匹配和排名以及他们的实现原理，在介绍这两个概念前先讲讲什么是索引。

### 2.4.1索引

“索引”这个词应该都很熟悉，通常出现在书的最后一个版块。通过索引你能够快速查找需要的概念，每一个概念都列出在书中具体出现的位置，比如页数。例如，“常熟理工556，223”的索引项，含义为：“常熟理工”这个词在书的第556，223页出现过。

下面我们假设有三个互联网的页面：

The cat sat on the mat

The dog sat stood the mat

The cat stood while a dog sat

三张网页的页码索引表如下：

表2-1 页面页码索引表

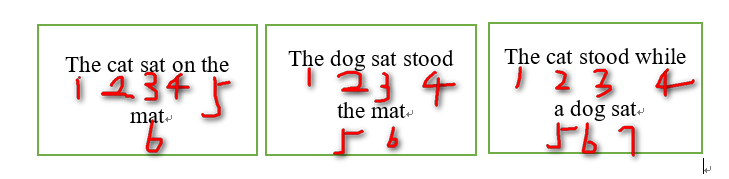
|  |  |  |  |
| --- | --- | --- | --- |
| a | 3 |  |  |
| cat | 1 | 3 |  |
| dog | 2 | 3 |  |
| mat | 1 | 2 |  |
| on | 1 | 2 |  |
| stood | 2 | 3 |  |
| the | 1 | 2 | 3 |
| while | 3 |  |  |

通过分析上表，可以发现a在第三张页面中出现过，the则表示在上述页面的第1，2，3张表中出现。通过这种方式，在搜索cat 、on、 while等单词时，搜索引擎可以非常快速的定位到。然而在实际搜索中，我们通常不会一个词一个词的搜索，而是词语或词组搜索。

比如我现在想要搜索cat sat（无引号）和“cat sat”（有引号）。第一项中cat sat，搜索引擎可以快速定位到cat和sat分别在第1、3页出现过。然后搜索“cat sat”，搜索引擎会优先搜索含有cat 和sat的页面，结果可以发现目标出现在第1页和第3页，这时，搜索就卡住了，因为它不知道具体的顺序是如何？谁在前谁在后？

如何解决这个问题？利用“词把戏”解决。

解决思路：索引不只是存储页码，而且还要存储这个词在页面在内出现的位置比如：第3个词的位置是3，第29个词的位置是29。



三张网页的新页码索引表如下：

表2-2 新页面页码索引表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | 3-5 |  |  |  |  |
| cat | 1-2 | 3-2 |  |  |  |
| dog | 2-2 | 3-6 |  |  |  |
| mat | 1-6 | 2-6 |  |  |  |
| on | 1-4 | 2-4 |  |  |  |
| sat | 1-3 | 3-7 |  |  |  |
| stood | 2-3 | 3-3 |  |  |  |
| the | 1-1 | 1-5 | 2-1 | 2-5 | 3-1 |
| while | 3-4 |  |  |  |  |

通过上表可以发现a出现在了第3页第5个位置，而cat则出现在第1页的第2个位置，第3页的第2个位置，其他单词以此类推。

当开始寻找词语“cat sat”,通过查看索引表中信息("cat":位置：1-2 3-2，“sat”:位置：1-3   3-7)。那么我们需要知道他们是否在同一个页面，3-2=1而且7-2=5.便知道了"cat sat"他们在第一页面位置相差1，所以时连在一起的词。通过这种方式便解决了词组的搜索。

### 2.4.2 匹配

1. 什么是匹配

假设我下载搜索的是“常熟理工学院4路车时刻表”，那么匹配就是在整个互联网中搜索包含这个字段的网页。所以说匹配阶段就是回答了“那些网页与我的搜索要求相符合”这个问题。具体实现如下：

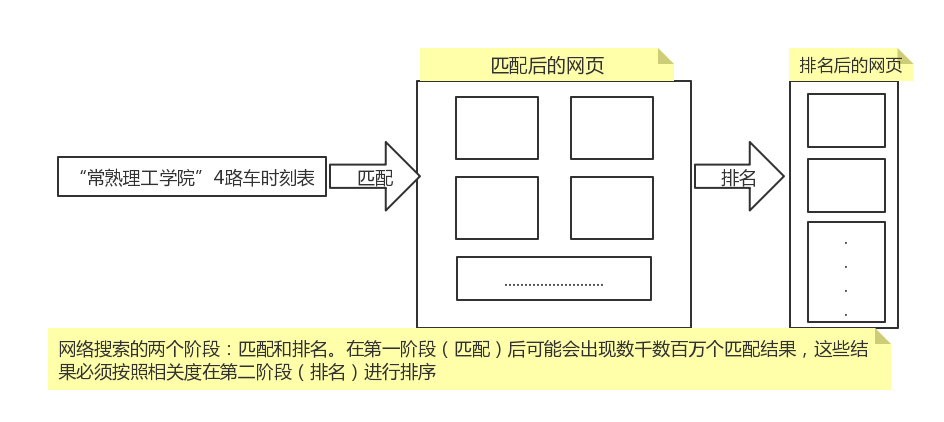


图2-3 匹配与排名流程

1. 为什么匹配

互联网中的资源是特别特别多的，用户很难从互联网中精确的找到自己想要搜寻的内容。为了提高搜索的准确度，每个页面需要有与之匹配的字段提供给用户，从而提高查询的效率和命中率。

### 2.4.3排名

1. 什么是排名

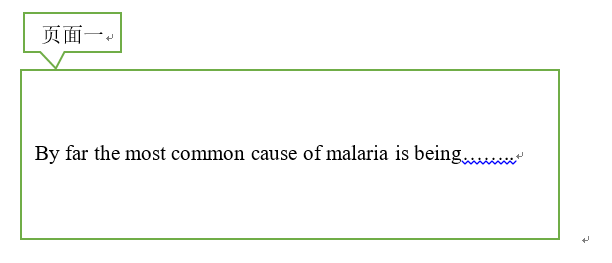
用生活中的例子来解释，考试排名、体测排名等等。和生活中的这些实例一样，区别就是互联网中的排名我们看不到，通常这项工作是由搜索引擎来完成。

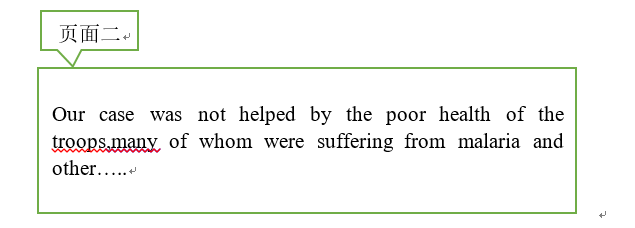
1. 为什么要排名

排名的目的是为了将搜索关键字命中最高的页面展示给用户。“排名”对于一个高质量、高效率的搜索引擎是至关重要的。

1. 如何排名

关于如何排名，通过例子说明：





加入我们要搜索“cause malaia”,根据3.3.1中可以知道case 1-6 2-2，malaia 1-8，2-19。根据页面中的含义，页面一是真正的原因，但是两个页面同事命中，这个时候又如何分辨呢？

网页结构是使用HTML来编写的，其中包含着大量的标签(tag)，通过分析这些标签能够做到简单的搜索功能。一般情况一篇文章的标题能够体现这篇文章的中心大意，所以提取标题的文字便成为了关键内容，这比起在文章中出现一两个单词更具有搜索价值。

一下为三段代码分别对应三张页面：

第一张：

1. **<titleStart>**my cat**</titleEnd>**
2. **<bodyStart>**the cat sat on the mat**</bodyEnd>**

第二张：

1. **<titleStart>**my dog**</titleEnd>**
2. **<bodyStart>**the dog stood on the mat**</bodyEnd>**

第三张：

1. **<titleStart>**my pets**</titleEnd>**
2. **<bodyStart>**the cat stood while a dog sat**</bodyEnd>**

其中我们可以得到标签的索引

表2-3 网页页码索引表

|  |  |  |  |
| --- | --- | --- | --- |
| </bodyEnd> | 1-12 | 2-12 | 3-13 |
| <bodyStart> | 1-5 | 2-5 | 3-5 |
| </titleEnd> | 1-4 | 2-4 | 3-4 |
| <titleStart> | 1-1 | 2-1 | 3-1 |

其他类推。

接下来，我们找到标题中有dog的网页。我们可以得到需要用到的索引：dog：2-3 2-7 3-11，<titleStart>  1-1   2-1   3-1, <titleEnd> 1-4  2-4 3-4。

首先搜索引擎会在索引表中查找以“1-”开头的索引，根据HTML标签相关的知识能够知道，在start以及end之间的内容便是我们要查找的。例如：1-1 1-4，然后搜索dog中的索引，没有“1-”开头的，加入没有则往下继续搜索索引值为2-1  2-4  ，再搜索dog中，发现有“2-"开头，那么判断"2-?"中的?是否在1<x<4范围之间，发现3处于1~4之间，那么便命中，依次往下搜索。注意的是：搜索引擎每次都只需要查询小部分索引项，便可以回应查询，除此之外搜索引擎只需遍历每个索引项一次。

# 3 Scrapy在校园网站搜索中的应用

## 3.1 Scrapy介绍与使用

### 3.1.1 Scrapy简介

Scrapy是Python开发的一个快速、分布式、易扩展、高效率的屏幕抓取和web抓取框架[7]。通常用于抓取web站点并从页面中精确出提取结构化的数据。Scrapy的应用非常广泛，在数据挖掘、信息处理或者存储历史数据等一系列程序中发挥重要作用。

Scrapy采用了Twisted异步技术来处理网络通讯信息的爬虫框架[8]。整体结构大致如下图所示：

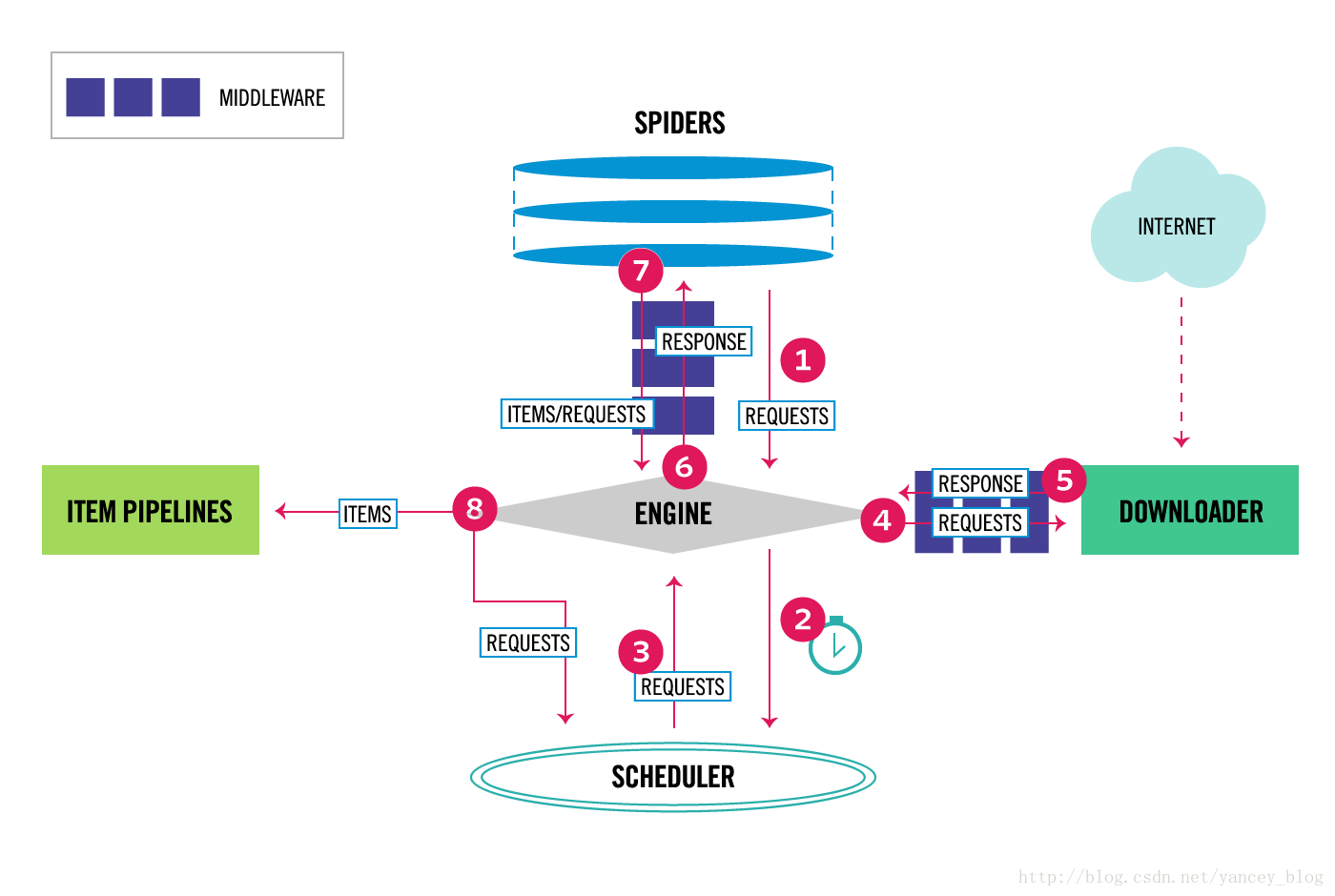


图3-1 Scrapy工作流程

**Scrapy爬虫框架主要包含了以下组件：**

1. 引擎(Engine) ：用来处理整个系统的数据流和触发事务
2. 调度器(Scheduler) ：接受引擎传递的请求, 压入队列中,当引擎再次请求时返回.。其功能可以想像成一个URL的优先队列, 由它来决定下一个要抓取的网址是什么, 同时肩负着除重复网址的功能
3. 下载器(Downloader) ：下载器用于下载网页内容, 并将网页内容返回给爬虫，Scrapy的下载器是建立在异步技术Twisted的基础之上的
4. 爬虫(Spiders) 爬虫可以说是整个系统的核心，爬虫的主要功能就是从网页中爬取到自己需要的内容
5. 项目管道(Pipeline) ：管道的作用是实体的持久化、清除不需要的内容、验证有效性。当页面被解析后会经过管道，可以经过几个特定次序处理数据。
6. 下载器中间件(Downloader Middlewares) ：下载中间件是位于Scrapy引擎和下载器之间的框架，这个框架是用来处理Scrapy引擎与下载器之间的请求及响应。
7. 爬虫中间件(Spider Middlewares) ：爬虫中间件介于Scrapy引擎和爬虫之间的框架，它的作用是用来处理蜘蛛的响应输入和请求输出。
8. 调度中间件(Scheduler Middewares) ：调度中间件介于Scrapy引擎和调度之间的中间件，处理从Scrapy引擎发送到调度的请求和响应。

### 3.1.2 Scrapy的使用

1. 创建项目：通过cmd进入想要防止项目的目录，在命令行中输入：
2. F:\python-wokspace>scrapy startproject ArticleSpilder
3. 新建后的项目目录如下：

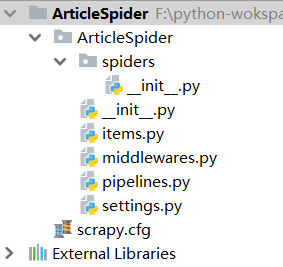


图3-2 Scrapy项目结构

1. 项目结构详解

* scrapy.cfg：项目的配置信息文件。
* item.py：定义数据存储模板，用于结构化存储数据，其作用类似于Django的Model。
* pipelines.py：数据处理行为，如：一般结构化的数据持久化
* settings.py ：配置文件，可在改文件中配置递归层数，并发数，自定义中间件等
* spiders ：爬虫目录，其内存放爬虫文件。

## 3.2 Scrapy中常见的问题

### 3.2.1 URL去重策略

1. 将访问过的url全部保存到数据库中

获得正在访问的url，将其存储到数据库中，在进行下次访问时从数据库中查询当前访问的url是否已经存储在数据库中。如果存在，这跳过这个url的访问。

但是这种方式每次访问都会从数据库查询一次，效率非常低，但这是最简单的一种方式，一般不采用。

1. 将访问过的url保存到set中

这种方式时利用python中set的不可重复的特性，通过python的set将url存储到内存中。这种方式虽然可以避免重复，但是占用的内存会越来越大，我们可以来进行下简单的计算。

假设内存中的url有一亿条，其中每个url有50个字符，Python采用的是Unicode编码，每个字符占两个字节，那么我们可以通过计算出得占用的内存数为：

http://chart.googleapis.com/chart?cht=tx&chl=%5Cfrac%7B100000000%5Ctimes%202byte%5Ctimes%2050%7D%7B1024%5Ctimes%201024%5Ctimes%201024%7D%5Capprox%209G

根据计算可得，内存的占用数大概为9G，占用的内存还是非常大的。

1. url经过md5等方法摘要之后保存到set中

根据上一个方法可以知道，通过set的方式直接保存url是非常占用内存的。而这种方式是对上一种方式的优化。

通过md5摘要之后的数据有固定的长度128bit，即一条url占用16字节。上一种方法一条url占用的是100字节，大概是这种方法的6倍。所以这种方法一亿条url占用大概1.5G的内存，还是可以接受的。在Scrapy框架中，我们通常使用的去重策略就是这一种。

1. 用bitmap方法，把访问过的url通过hash函数哈细化后映射到某一位

上图所示为计算机存储单位的一个字节(byte)，即8位(bit)。每个位有0，1两种状态。在这种情况下，我们将url经过hash化之后对应的是内存中的一块地址，我们根据当前这位地址状态，如果为1表示已经存在，如果为0表示不存在。通过这种方法能够节省相当大的空间。

但这种方法会遇到许多问题，如多条url映射到同一地址，会产生冲突。

1. bloomfilter方法对bitmap进行改进，擦用多重hash函数来降低冲突

通过对上一种方法的分析，我们可以知道在通过bitmap去重时会有遇到地址冲突的问题。而bloomfilter就是对这种方法进行的改进。bloomfilter通过多次hash映射，来降低冲突的概率[9]。

### 3.2.3 关于反爬虫以及应对反爬虫的策略

1. 通过Headers反爬虫

从用户请求的Headers(请求头)入手是目前大多数网站采用最多的反爬虫策略。很多网站都会通过对Headers的User-Agent进行检测，来判断是否是通过浏览器及西宁访问，除此之外还有一部分网站会对Referer进行检测。

相对来说，这类反爬虫措施比较好解决。想要应对这种反爬虫措施，开发者需要在爬虫的请求中添加Headers（请求头），即将浏览器的User-Agent添加到爬虫的Headers中；

1. 基于用户行为反爬虫

除了上述反爬虫措施，还有一部分网站是通过检测用户行为来进行反爬虫，例如同一IP在很短的时间内多次访问过同一页面，或者是同一账户在很短时间内进行了大量重复操作。

对于这种反爬虫，使用IP代理能够解决。专门定义一个爬虫，爬取网上公开的代理ip（比如西刺IP代理），将其存储起来。在我们访问时每隔一段时间或者进行一定数量操作后更换IP，模拟出多个用户访问的效果，欺骗被爬取的网站。

1. 动态页面的反爬虫

上述的几种情况是比较简单的反爬虫措施。但是由于网站架构的演变，很多网站的数据以ajax方式请求，也有的是通过JavaScript生成的。对与这种情况，首先用fildder或chrome插件进行分析。如果能够提取到到ajax请求，并且能公众分析出具体的参数和响应的含义，那我们就能刚才提到的方法，通过使用requests或者urllib3来认为构造ajax请求，并对响应的json数据进行分析、提取。

然而有些网站会把ajax请求的所有参数先加密再传输。这样很难通过上述方法构造请求。那该怎么解决呢？此时我们就不能用上面的方法了。这种情况我们可以使用selenium+phantomJS框架，通过调用浏览器内核，利用phantomJS来执行js。使用这种方式，从填写表单到点击按钮甚至是滚动页面，我们可以全部模拟。

用这种方式几乎能够应对所有的反爬虫措施，这种方式不是伪装成浏览器来获取数据，而是它本身就是一个浏览器。phantomJS可以说是一个没有界面的浏览器，和我们使用的浏览器区别是这个浏览器不是用户操作。使用这种策略可以做很多事，比如识别点触式验证码、滑动式的验证码、对页面表单暴力破解等等。

1. 常见的反爬虫与应对如下图所示：

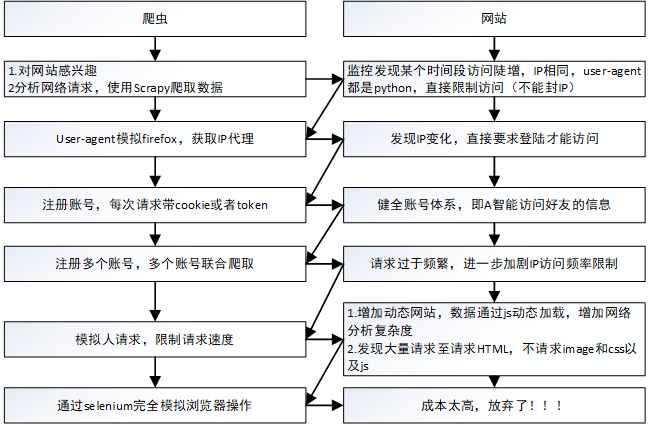


图3-3 反爬虫与应对

# 4 校园搜索引擎系统的设计与实现

在上一章我们对Scrapy的原理和一些常见的问题进行了讲解，接下来将进入系统的开发阶段。通过Scrapy爬取数据并结合ElasticSearch和Django完成整个系统的开发。

## 4.1 系统总体架构设计

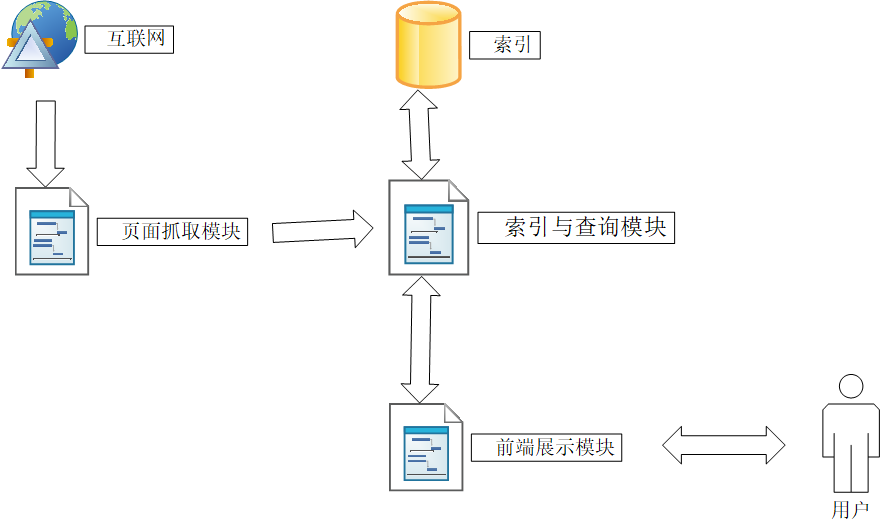


图4-1 系统总体架构设计

如图5.1所示，系统共分为三大模块，页面抓取模块，索引与查询模块，页面展示模块。其中页面抓取模块负责从互联网中抓取网页信息，索引与检索模块将网页抓取模块得到的网页信息，通过分词插件进行分词后添加到索引，用户从前端展示界面中输入查询的关键字，然后点击搜索，索引与查询模块会根据查询关键字进行检索排序，然后将结果返回给页面展示模块，用户便会得到想要的内容进行浏览。其中具体实现接下来一一讲解。

## 4.2 “常熟理工学院新闻网”内容抓取模块

爬虫模块时通过Scrapy框架实现的，在上一章节已经介绍了Scrapy框架的基本知识，安装，工程的创建等。接下来将一次为基础，爬取校园网中的新闻内容。

### 4.2.1 爬取流程分析

获取列表页url

Request进行下载

提取文章中的字段

获取下一页url

Callback=parse()

下一页是否存在

图4-2 爬虫爬取流程

爬虫的实现主要用到的时递归的思想，首先介绍下爬虫文件中的的一些参数、方法的作用。

|  |
| --- |
| class CslgNewsSpider(scrapy.Spider):  name = 'cslg\_news'  allowed\_domains = [' news.cslg.edu.cn ']  start\_urls = [http://news.cslg.edu.cn/Index/newslist/id/17/p/  1.html ']  def parse(self, response):  print('我进入了解析器') |

1. name，定义的时爬虫的名字。
2. allowed\_domains，可选内容，包含了爬虫(spider)允许爬取的域名列表。
3. start\_urls，URL列表。其中定义了爬虫爬取的顺序。
4. parse(response)，作为Scrapy框架中处理下载的response的默认方法。

### 4.2.2 新闻页面分析

编写爬虫时需要根据具体的页面来分析每个页面的爬取规则。接下来，以常“熟理工学院新闻网”为模板，对页面进行分析。

在阅读一篇新闻时，我们通常会注意到标题、内容、发布日期这些信息。所以，我们需要存储的也是这三个简单的字段。接下来就是需要从新闻的源代码中提取出来。Scrapy提供了Xpath和CSS两种提取规则。本系统我们采用的时CSS的提取规则。“常熟理工学院新闻网“页面如下图所示。



图4-3 “常熟理工新闻网“页面

新闻网首页共有六个模块，“学校要闻”，“通知通告”，“理工校报”，“思想前行”，“理工星空”，“学术报告”。经过分析，这六个模块页面的源码结构时一样的，所以我们接下来的分析都以“学校要闻”模块为示例。

首先我们为了能够获取到所有的新闻，所以要获取到下一页按钮中的“href”参数的值来下载下一页的内容，并且还要获取到每一条具体新闻的url来实现新闻详情的读取。

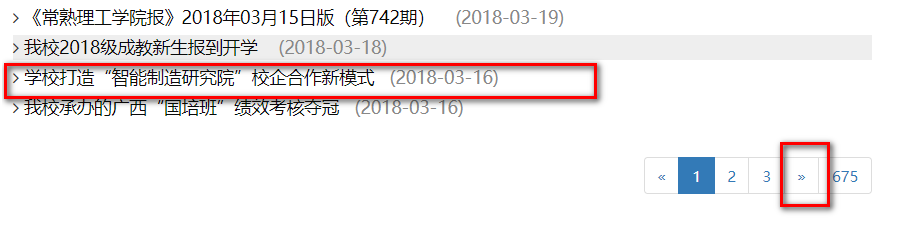


图4-4 新闻下一页按钮和新闻列表

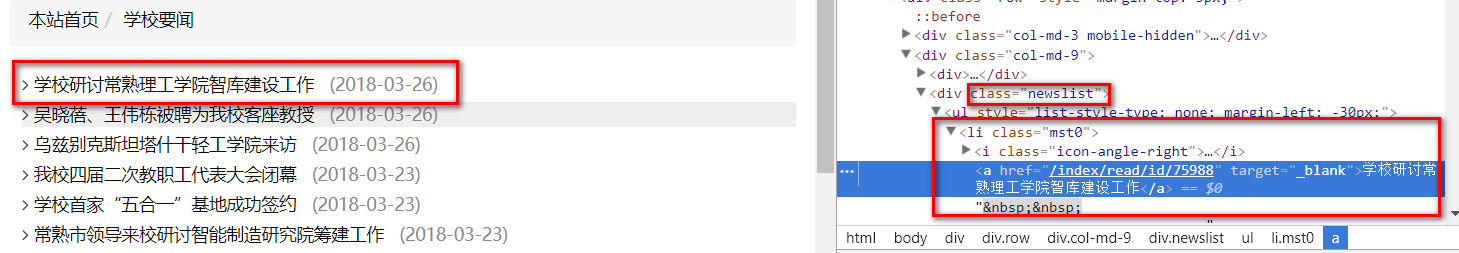


图4-5 新闻列表及源码

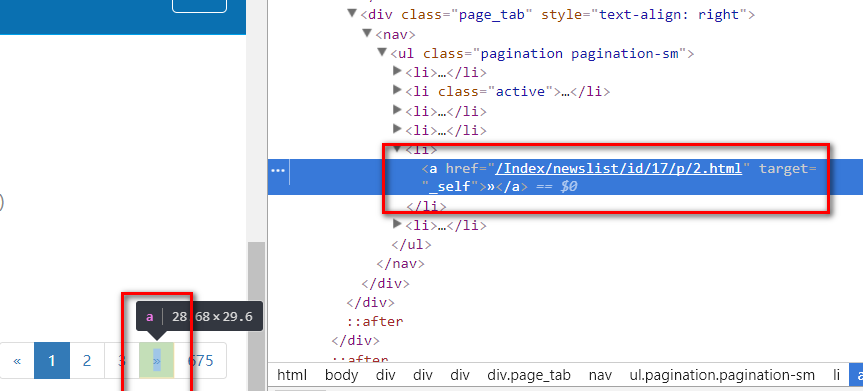


图4-6 下一页按钮及源码

同过对页面的分析，我们使用CSS选择器的方式来进行数据的提取。在Scrapy中，提供了ItemLoader填充机制，实现对Item的值的填充。通过分析，所有的新闻列表都在<li class=“mst0”></li>标签下的<a>标签下，因此我们提取出所有的<a>标签的href参数获得的就是当前页的新闻列表。经过分析，下一页的链接也是a标签，而且链接的下一页是以“.html”结尾的，并且下一页一直是在倒数第二个位置，所以便很容易提取出这些参数。

**部分代码如下：**

|  |
| --- |
| # 解析列表页中的所有新闻url并交给scrapy下载后并进行解析  post\_nodes = response.css(".mst0 a::attr(href)").extract()  for post\_node in post\_nodes:  yield Request(url=parse.urljoin("http://news.cslg.edu.cn", post\_node), callback=self.parse\_detail)  # 提取下一页并交给scrapy进行下载  page\_urls=response.css("a[href$='.html']::attr(href)").extract()  next\_url = page\_urls[len(page\_urls)-2] |

提取到列表和下一页之后，我们要做的是进一步提取新闻中的详细内容。刚才已经分析了，我们要获得的内容分别为标题，创建日期，新闻具体内容。我们首先来看下标题。

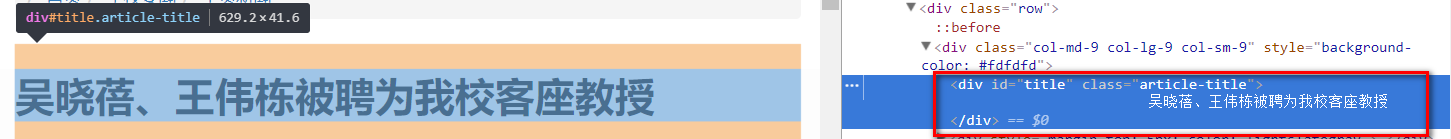


图4-7 新闻标题及源码

标题是在一个class=“title”的<div>下，同过CSS元素选择器我们可以很容易得到标题的内容。接下来看创建日期和新闻详情的代码。



图4-10 创建日期及其源码

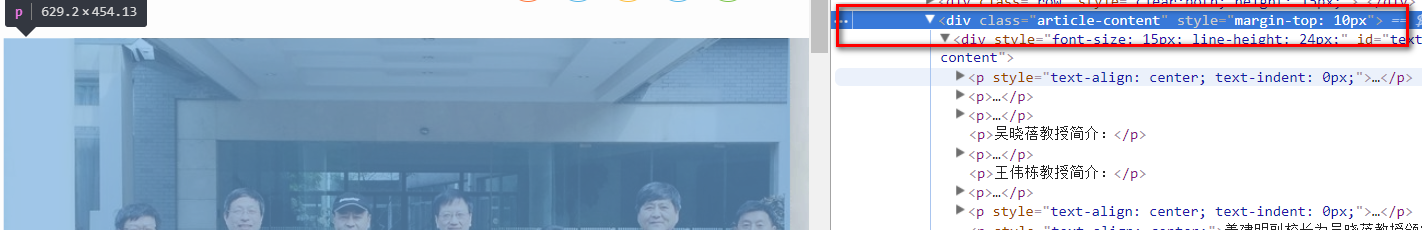


图4-8 新闻内容及其源码

通过以上分析我们很容易发现特点，创建日期所在的区域是一个id为date-topic的<div>我们通过CSS选择器能够很容易得到，同理新闻的详情内容页很容易得到，对于内容部分，我们获取的是整个<div>中的内容，其中包括了很多HTML标签，这些我们通过之后使用正则表达式进行过滤即可。

因为我们搜索引擎的效果需要跳转到具体的页面，所以我们需要保存下新闻详情的url以提供下次访问。为了实现url去重的策略，我们采用本文中提到的url经过MD5摘要之后的策略进行去重，所以也要保存下url经过MD5摘要后的值。

**具体代码如下：**

|  |
| --- |
| def parse\_detail(self, response):  # 实例化  article\_item = JobBoleArticleItem()  # 通过item loader加载item  item\_loader = ArticleItemLoader(item=JobBoleArticleItem(), response=response)  #因为校园网上有很多链接到外部资源的新闻，比如微信公众号文章、#大国重器央视视频等，所以对不同资源的提取方式不同  #对于校内发布的信息格式固定，采用一下规则提取  if response.url.find("/Index/newslist") or response.url.find("/index"):  # 通过css选择器将后面的指定规则进行解析  item\_loader.add\_css("title", ".article-title::text")  item\_loader.add\_value("url", response.url)  item\_loader.add\_value("url\_object\_id", get\_md5(response.url))  item\_loader.add\_css("create\_date", "#date-topic::text")  item\_loader.add\_css("content", ".article-content")  # 调用这个方法来对规则进行解析生成item对象  article\_item = item\_loader.load\_item()  #如果是外部资源，因为规则不固定，所以只提取标题和内容，其中##标题提取<title>标签中的，content则直接提取正文的标签之间的内容  else:  item\_loader.add\_css("title", "title::text")  item\_loader.add\_value("url", response.url)  item\_loader.add\_value("url\_object\_id", get\_md5(response.url))  item\_loader.add\_css("create\_date", "title::text")  item\_loader.add\_css("content", "body::text")  # 已经填充好了值调用yield传输至pipeline  yield article\_item |

## 4.3 “计算机科学与工程学院”内容抓取模块

有了上一小节对“常熟理工学院新闻网”的爬取经验，接下来我们对“计算机科学与工程学院”新闻通知进行爬取。首先我们来分析下入口页面的情况。具体如下图所示：

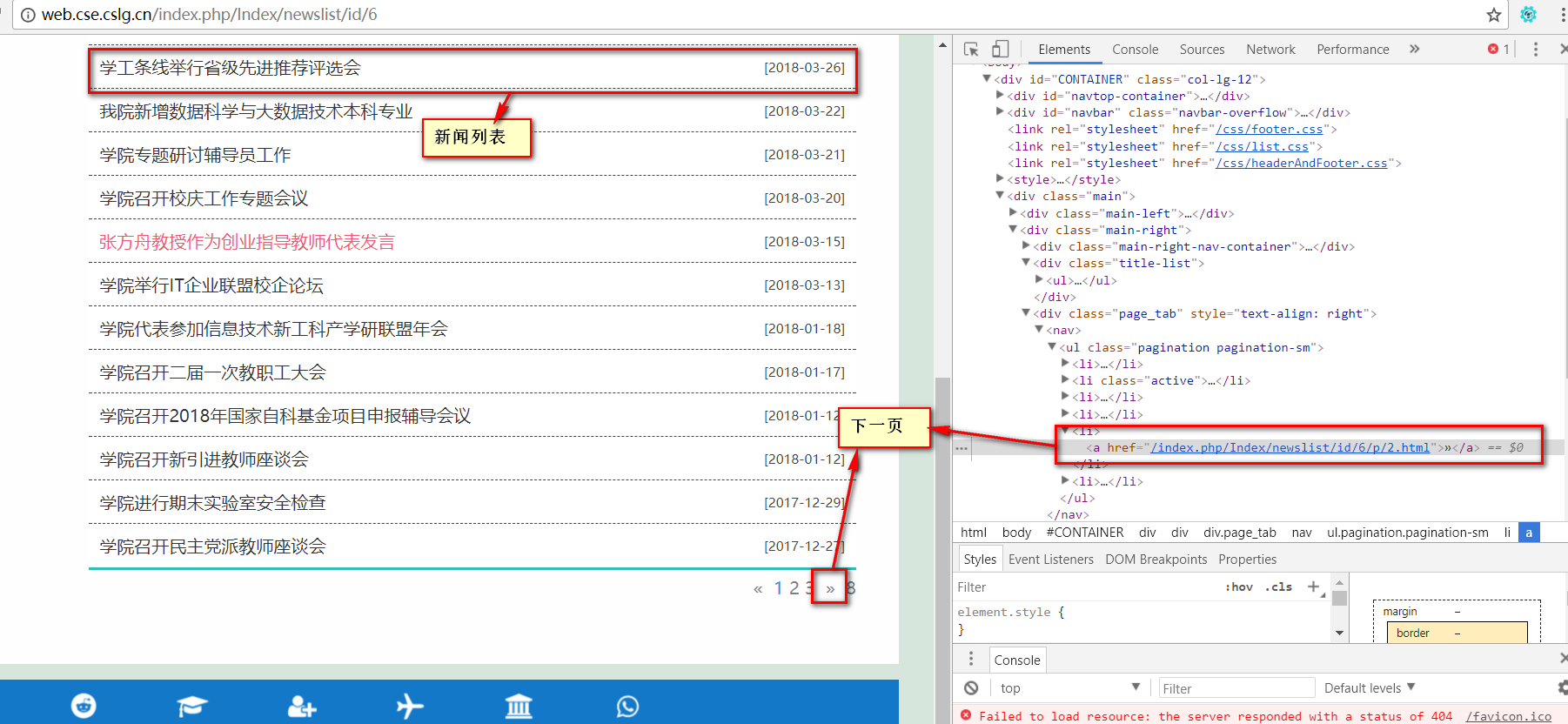


图4-9 “计算机科学与工程学院”页面展示

根据对“计算机学院”新闻通知页面的分析可以发现，页面的布局、标签属性等与上一节的“常熟理工新闻网”页面基本一致。此时，我们只要套用上一节的代码便可以实现对每一页进行递归读取。

**关键代码如下：**

|  |
| --- |
| # 解析列表页中的所有新闻url并交给scrapy下载后并进行解析  post\_nodes = response.css(".mst0 a::attr(href)").extract()  for post\_node in post\_nodes:  yield Request(url=parse.urljoin("http://news.cslg.edu.cn", post\_node), callback=self.parse\_detail)  # 提取下一页并交给scrapy进行下载  page\_urls=response.css("a[href$='.html']::attr(href)").extract()  next\_url = page\_urls[len(page\_urls)-2] |

因为两个站点要爬取的内容都是新闻资讯，所以我们对其保存的结构是一致的，都是标题、创建时间、内容这三个部分。“计算机科学与工程学院”的新闻具体内容和源码如下所示：

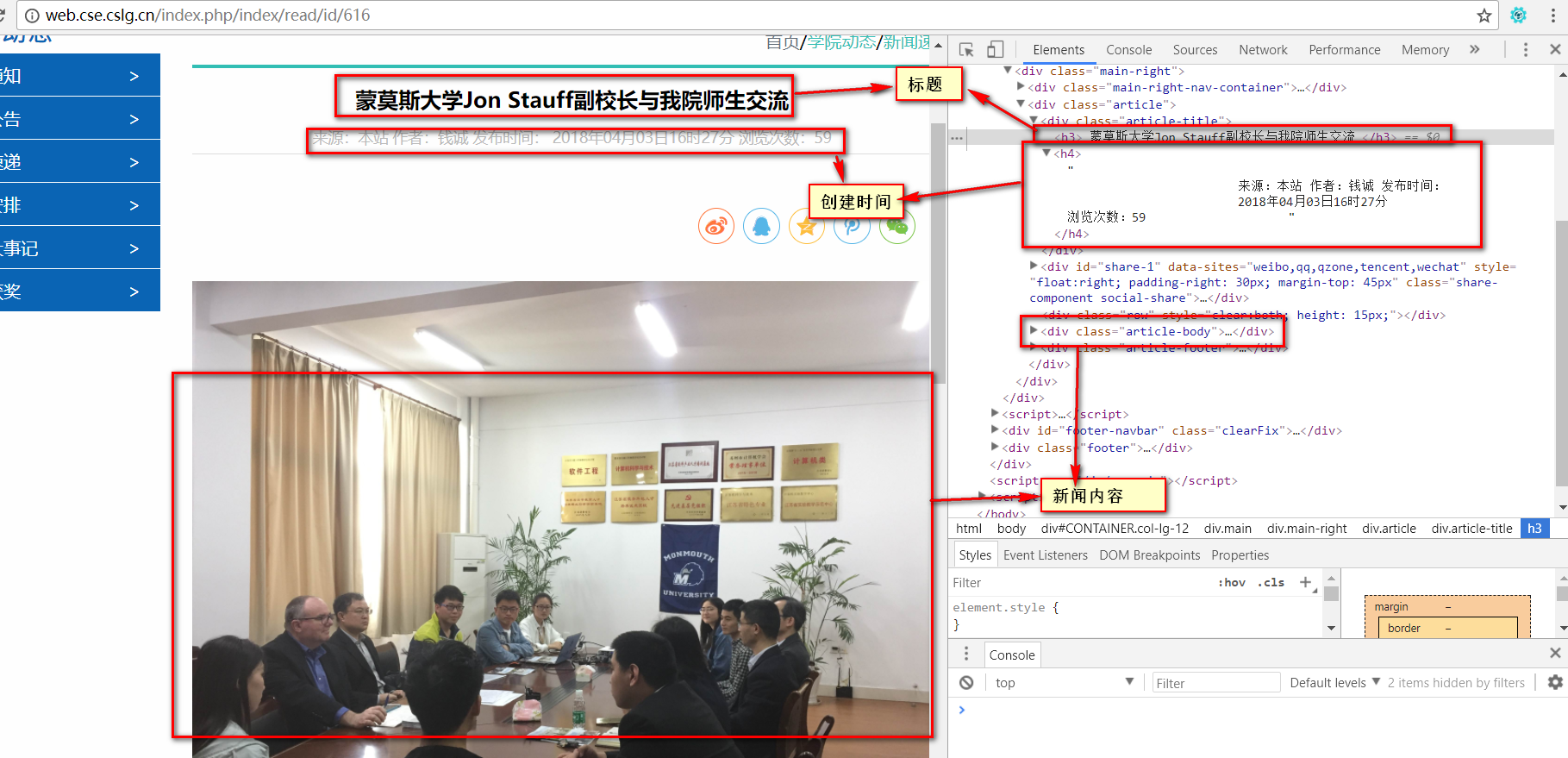


图4-10 “计算机学院”新闻详情页

通过对以上源码的分析发现，“计算机科学与工程学院”的新闻详情页与“常熟理工新闻网”的页面结构有很大的不同。在“常熟理工学院新闻网”中，页面的标题是在一个class = article-title的<div>标签中，但是在“计算机科学与工程学院”的页面中，是一个<h3>标签中的内容，此时我们只需要使用CSS中元素选择器就可以获得内容，其他的两个字段同理。

**关键代码如下：**

|  |
| --- |
| def parse\_detail(self, response):  # 实例化  article\_item = JobBoleArticleItem()  # 通过item loader加载item  item\_loader=ArticleItemLoader(item=JobBoleArticleItem(),response=response)  #通过对页面分析发现该站点没有引用外部资源所以不考虑不符合规则  #情况  if response.url.find("/Index/newslist") or response.url.find("/index"):  # 通过css选择器将后面的指定规则进行解析。  item\_loader.add\_css("title", "h3::text")  item\_loader.add\_value("url", response.url)  item\_loader.add\_value("url\_object\_id", get\_md5(response.url))  item\_loader.add\_css("create\_date", "h4::text")  item\_loader.add\_css("content", ".article-body")  # 调用这个方法来对规则进行解析生成item对象  article\_item = item\_loader.load\_item()  yield article\_item |

当需要注意的是，因为“计算机科学与工程学院”新闻详情中的日期，是和其他信息冗杂在一起的，数据格式为：“来源：本站 作者：钱诚 发布时间：2018年04月03日16时27分浏览次数：59”。为了提取出需要的信息，我们需要对数据进行清洗。

关键代码如下：

|  |
| --- |
| def make\_data\_clean(self):  # 对日期格式进行清洗，提取出日期  # 来源：本站 作者：钱诚 发布时间：  # 2018年04月03日16时27分  # 浏览次数：44  #1 去除所有空格  new\_str = self["create\_date"].strip().replace("·", "").strip()  #获得“分”的下标  minute\_index = new\_str.find("分")  #获得“年”的下标  year\_index = new\_str.find("年")  # 进行截取，获得满足格式的日期  self["create\_date"] = new\_str[year\_index - 4:minute\_index + 1] |

此时，数据的爬取工作结束，下一小节将对爬取的数据进行格式化存储。

## 4.4 索引与检索模块

### 4.4.1 ElasticSearch介绍

ElasticSearch是一个基于Lucene开发的搜索服务器。它提供了一个分布式的多用户能力的全文搜索引擎，基于RESTful web接口[11]。Elasticsearch底层采用Java开发的，支持开源协议，是当前流行的企业级搜索引擎。目前Google,Facebook等多家公司在使用这款引擎，

### 4.4.2 ElasticSearch安装与配置

在windows下安装比较简单，在官放网站上下载需要的版本放到目录下进行解压即可，

在本系统使用的是elasticsearch-rtf版本。该版本是由国人推出的已经安装很多插件的版本，对中文支持良好，省去了很多插件安装的功能。

* ElasticSearch-rtf安装

1. 在github中搜索elasticsearch-rtf或直接输入https://github.com/medcl/elasticsearch-rtf进行下载
2. 下载完成后进行解压，进入bin目录下，运行elasticsearch.bat，待运行成功后打开浏览器输入127.0.0.1:9200出现如下界面则说明安装成功。

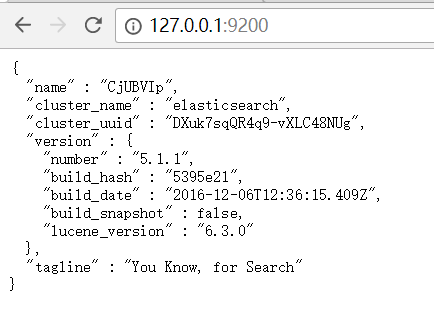


图4-11 运行成功图

* ElasticSearch插件安装

1. elasticsearch-head

head插件的功能类似于Navicat，基于浏览器用于管理数据库。需要注意的是在elasticsearch5.X之前head是以插件的形式集成到elasticsearch中的，在5.X版本之后则需要额外安装。因为本系统使用的是5.1.1的版本所以需要额外安装Node.js环境后才能正常使用head插件。因为网上有很多安装教程，所以这里不做具体讲解

1. kibana插件

Kibana是一款开源的数据分析与可视化平工具。Kibana设计初衷是与ElasticSearch配套使用，但随着发展可应用于其他数据可视化。可以通过kibana查看、添加、删除存放在Elasticsearch索引中的数据。

Kibana和Head插件一样基于浏览器，使你能快速创建、查看和分享Elasticsearch中的数据，Kibana的安装和使用非常方便 ，Kibana从官网下载与elasticsearch相同的版本后解压，进入bin目录下，运行kibana.bat在浏览器输入127.0.0.1:5601即可使用。



图4-12 运行结果图

### 4.4.3保存数据到ElasticSearch中

* 基础知识

在进行保存之前，我们需要知道ElasticSearch的一些基本概念，这样有助于接下来的操作。在这里将ElasticSearch中的一些概念与数据库中的概念的进行对比，有助于理解。

表4-1 概念对比

|  |  |
| --- | --- |
| ElasticSearch中概念 | 数据库中概念 |
| index | 数据库 |
| type | 表 |
| document | 记录 |
| field | 字段 |

除此之外，还要理解一些其他的概念：

1. 集群：一个或多个节点组织在一起
2. 节点：一个集群中的一台服务器
3. 分片：索引划分为多份的能力，可以水平分割，支持扩展容量，并且能够从多个分片响应请求
4. 副本：分片的的一份或多份，在使用中如果一个解点失败，其他解点可以顶上，保证系统正常使用。

* 保存数据

ElasticSearch提供了面向python的接口，目前有许多操作ElasticSearch的包，为系统开发提供了便利。

在ElasticSearch中提供了Mapping映射，创建索引之前，能够预先定义存储模型中的字段类型以及相关属性，每个字段只可以定义一种类型。在ElasticSearch中的属性比mysql里面丰富。除此之外，ElasticSearch能够根据json源数据来判断具体是什么基础类型。Mapping就是我们定义的字段的数据类型，通过这一操作能够告诉ElasticSearch如何索引数据和是否能够被检索。这样能够让索引建立的更加细致和完善，大多数情况下是我们不需要自己定义。

因为通过上两小节的分析可以发现，我们对“常熟理工学院新闻网”和“计算机科学与工程学院新闻通知”的存储结构设计的是一样的。所以我们定义保存的文档类型时可以选用相同的结构，保存到同一个索引中以减小工作量。关于新闻的来源我们可以使用URL进行分别，具体我们在4.5节中进行讲解。下面代码是我们自己定义的保持的文档类型。

**关键代码如下：**

|  |
| --- |
| # 与服务器进行连接，允许多个  connections.create\_connection(hosts=["localhost"])  # ElesticSearch存储新闻结构  class ArticleType(DocType):  #搜索建议  suggest = Completion(analyzer=ik\_analyzer)  #新闻标题，进行分词  title = Text(analyzer="ik\_max\_word")  #创建日期  create\_date = Text()  url = Keyword()  url\_object\_id = Keyword()  #新闻内容  content = Text(analyzer="ik\_max\_word")  #爬取日期  crawl\_time = Date()  class Meta:  index = "jobbole"  doc\_type = "article" |

其中title,content，create\_date都设置成了文本类型以便进行分词和索引。其中url和url\_object\_id则是keyword类型，不需要进行分词，能够被用来检索过滤、排序以及聚合。keyword 类型字段只能用本身来进行检索。crawl\_time为抓取日期，我们将其设为Date类型。

以上是定义的ElasticSearch中的存储模型，此时我们还需要定义爬虫的Items,用来保存从页面爬取的结构化数据：

|  |
| --- |
| # 常熟理工新闻网items类  class JobBoleArticleItem(scrapy.Item):  title = scrapy.Field()  create\_date = scrapy.Field(  )  url = scrapy.Field()  url\_object\_id = scrapy.Field()  content = scrapy.Field()  crawl\_time = scrapy.Field()  # 保存常熟理工新闻网新闻到es中  def save\_to\_es(self):  article = ArticleType()  article.title = self['title']  article.create\_date = self["create\_date"]  article.content = remove\_tags(self["content"])  article.url = self["url"]  article.meta.id = self["url\_object\_id"]  # 在保存数据时便传入suggest  article.suggest=gen\_suggests(es\_article,ArticleType.\_doc\_type.index,  ((article.title, 10), (article.content, 3)))  if redis\_cli.get("jobbole\_count"):  jobbole\_count = pickle.loads(redis\_cli.get("jobbole\_count"))  jobbole\_count = jobbole\_count + 1  jobbole\_count = pickle.dumps(jobbole\_count)  redis\_cli.set("jobbole\_count", jobbole\_count)  else:  jobbole\_count = pickle.dumps(JOBBOLE\_COUNT\_INIT)  redis\_cli.set("jobbole\_count",jobbole\_count)  article.save() |

这样，我们就成功进行了保存，保存的数据结果如下所示。



图4-13 部分保存的数据

到此为止，我们完成了数据的爬取、建立分词、索引等操作，接下来将进入用户交互界面的开发。

## 4.5 前端展示模块

Python的Web框架虽然说是百花齐放，但是Django可以说是众多框架中最好的。Django有强大的社区支持、完善的文档、较高的市场占有率、上手简单等等优势。相比其他框架，Django自带的ORM使用简单而且综合评价高于SQLAlchemy。除此之外，Django的使用用户多，第三方库最丰富，能为开发者节省时间，提高开发效率。综合考虑，本系统使用Django作为Web框架。

### 4.5.1 Django介绍

目前有多种基于Python的Web框架，Django是极具代表性的一种开源框架，它可以用来进行Web的快速开发。许多成功的网站和APP的开发都基于Django。

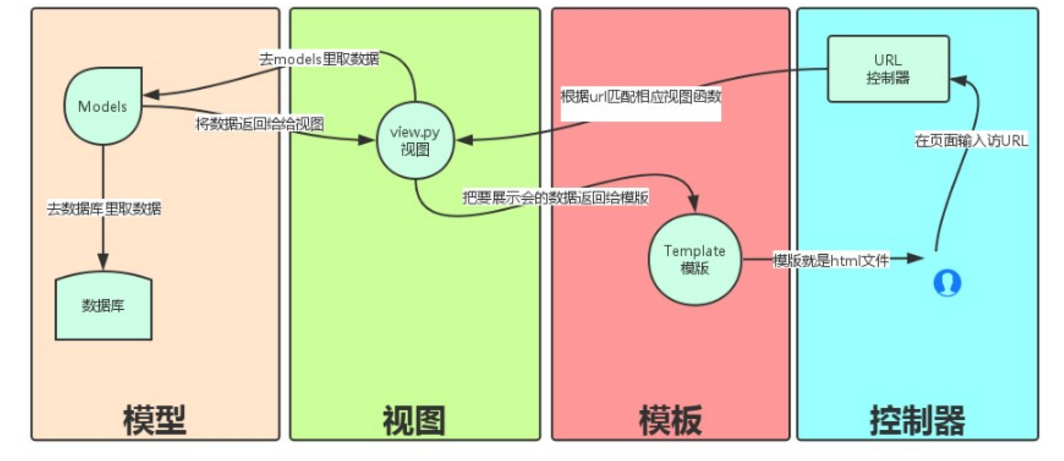


图5-14 django原理图

Django也是一个MVC框架（关于MVC架构的内容不做赘述）。但是在Django中，控制器接受用户输入的部分由框架自行处理，所以 Django 里更关注的是模型（Model）、模板(Template)和视图（Views），称为 MTV模式[12]：

M 代表模型（Model），即数据存取层。 该层处理与数据相关的所有事务： 如何存取数据、如何验证数据有效性以及数据之间的关系等。

T 代表模板(Template)，即表现层。 该层处理与表现相关的决定： 如何在页面以及其他类型的文档中进行显示。

V 代表视图（View），即业务逻辑层。该层主要是存取模型及调取恰当模板的相关逻辑。可以把它看作模型与模板之间的桥梁。

### 4.5.2 Django的使用

1. 将静态资源引入，在django中，我们需要配置静态资源的路径。

|  |
| --- |
| #settings.py  BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  STATIC\_URL = '/static/'  STATICFILES\_DIRS = [  os.path.join(BASE\_DIR,'static')  ]  MEDIA\_URL = '/media/'  MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media') |

1. 定义Model，需要和上一节保存的es结构一致。

|  |
| --- |
| # 与服务器进行连接，允许多个  connections.create\_connection(hosts=["localhost"])  class ArticleType(DocType):  #搜索建议  suggest = Completion(analyzer=ik\_analyzer)  #新闻标题，进行分词  title = Text(analyzer="ik\_max\_word")  #创建日期  create\_date = Text()  url = Keyword()  url\_object\_id = Keyword()  #新闻内容  content = Text(analyzer="ik\_max\_word")  #爬取日期  crawl\_time = Date()  class Meta:  index = "jobbole"  doc\_type = "article" |

1. 编写和配置view，当我们在views.py中编写完页面的处理逻辑后还需要在urls.py对试图配置路由。这个功能和SpringMVC中的前端控制器相似。下面为本系统配置路由视图代码。

|  |
| --- |
| urlpatterns = [  path('admin/', admin.site.urls),  path('', IndexView.as\_view(),name = "index"),  path('suggest/',SearchSuggest.as\_view(),name = "suggest"),  path('search/', SearchView.as\_view(), name="search"),  path('favicon.ico', favicon\_view),  ] |

### 4.5.3 搜索引擎的开发流程

经过以上简单的配置之后，我们从搜索引擎的工作流程入手，对系统进行设计与开发。在第二章中我们已经介绍了搜索引擎的工作流程，并且第二章已经完成的内容的爬取、分词、索引等操作。接下来，我们将继续以上步骤进行开发。

用户使用搜索引擎时，第一步是在文本框中输入想要查询的关键字。关于文本框页面的代码如下所示。

|  |
| --- |
| <div class="inputArea">  <input type="text" class="searchInput" />  <input type="button" class="searchButton" onclick="add\_search()" />  <ul class="dataList">  <li></li>  </ul>  </div> |

通过对以上代码的分析可以知道，当我们输入完关键字点击搜索按钮时会出发事件，相关JS代码如下所示

|  |
| --- |
| var search\_url = "{% url 'search' %}"  function add\_search(){  var val = $(".searchInput").val();  if (val.length>=2){  //点击搜索按钮时，去重  KillRepeat(val);  //去重后把数组存储到浏览器localStorage  localStorage.search = searchArr;  //然后再把搜索内容显示出来  MapSearchArr();  } window.location.href=search\_url+'?q='+val+"&s\_type="+$(".searchItem.current").attr('data-type')} |

在以上代码的最后一行可以发现，当点击搜索按钮时，JS代码会执行相应的内容。即携带关键字作为参数，向后台请求，请求的视图为“search“。根据上一小节最后我们对路由的配置可以知道，“search”视图对应SearchView。所以我们需要在views.py中完成我们的搜索类——class SearchView 。

首先，我们需要从请求中提取出用户搜索的关键字信息。

|  |
| --- |
| class SearchView(View):  def get(self, request):  key\_words = request.GET.get("q", "")  # 通用部分  # 实现搜索关键词keyword加1操作  redis\_cli.zincrby("search\_keywords\_set", key\_words)  # 获取topn个搜索词  topn\_search\_clean = []  topn\_search = redis\_cli.zrevrangebyscore(  "search\_keywords\_set", "+inf", "-inf", start=0, num=5)  for topn\_key in topn\_search:  topn\_key = str(topn\_key, encoding="utf-8")  topn\_search\_clean.append(topn\_key)  topn\_search = topn\_search\_clean |

当得到用户请求的关键字后，我们需要从索引库中检索用户关键字信息。为了更好的显示结果，我们对命中词进行高亮显示。

|  |
| --- |
| response = client.search(  index="jobbole",  request\_timeout=60,  body={  "query": {  "multi\_match": {  "query": key\_words,  "fields": ["tags", "title", "content"]  }  },  "from": (page - 1) \* 10,  "size": 10,  "highlight": {  "pre\_tags": ['<span class="keyWord">'],  "post\_tags": ['</span>'],  "fields": {  "title": {},  "content": {},  }  }  }  ) |

因为本系统目前针对了两个站点，即“常熟理工学院新闻网”和“计算机科学与工程学院新闻通知”为对象。在显示结果时我们需要对新闻的来源加以区分。这里我们通过URL的方式进行区分，因为“常熟理工学院新闻网”和“计算机科学与工程学院”分属于不同的域名，“常熟理工学院新闻网”是“news.cslg.edu.cn”而“计算机科学与工程学院”是“web.cse.cslg.cn”。所以，我们通过对来源的URL进行判断，如果包含“news.cslg.edu.cn”则是来自“常熟理工学院新闻网”否则则是“计算机科学与工程学院”。具体代码如下：

|  |
| --- |
| hit\_dict["url"] = hit["\_source"]["url"]  if (hit\_dict["url"].find("news.cslg.edu.cn") >= 0):  hit\_dict["source\_site"] = "常熟理工学院新闻网"  elif (hit\_dict["url"].find("web.cse.cslg.cn") >= 0):  hit\_dict["source\_site"] = "计算机科学与工程学院新闻中心" |

ElasticSearch查询的结果为Json类型，而我们的前后台数据的交互也是Json格式，所以我们将结果添加到response中作为返回。

|  |
| --- |
| return render(request, "result.html", {"page": page,  "all\_hits": hit\_list,  "key\_words": key\_words,  "total\_nums": total\_nums,  "page\_nums": page\_nums,  "last\_seconds": last\_seconds,  "topn\_search": topn\_search,  "jobbole\_count": jobbole\_count,  "s\_type": s\_type,  }) |

在result.html页面对结果进行显示，以下是对来源和发布时间的显示。

|  |
| --- |
| <span class="fileType">  <span class="label">来源：</span>  <span class="value">{{ hits.source\_site }}</span>  </span>  <span class="dependValue">  <span class="label">得分：</span>  <span class="value">{{ hits.score }}</span>  </span> |

### 4.5.4 搜索引擎用户界面与结果展示

接下来我们以“常熟理工学院 发展战略”为关键词进行查询。查询结果如下：



图4-15 查询结果一

我们在以“蒙莫斯大学Jon Stauff副校长”为关键字进行查询，查询结果如下：



图4-16 查询结果二

查询结果以关键字命中数量进行平分和排序，对关键字进行标红。通过查询的结果我们很容以得到我们想要得到的信息。

至此，“常熟理工学院校内搜索引擎”开发完成。

# 5 系统测试

## 5.1 测试环境

测试环境为：

1. 系统环境：window10 专业版 64位
2. 软件环境：python3.6，chrome
3. 硬件环境：cpu-i5-4200u双核处理器主频1.7G，4G内存

## 5.2 测试结果分析

接下来，我们以“常熟理工学院 发展战略”位关键字分别在“常熟理工学院校园搜索引擎”、“百度搜索引擎”和“搜狗搜索引擎”分别进行搜索，然后进行比较。得到结果如下图所示。



图5-1 搜狗搜索结果



图5-2 百度搜索结果



图5-3 校园搜索引擎搜索结果

通过以上结果，我们可以发现。百度和搜狗搜索引擎虽然也会对校园新闻进行抓取，但是仅仅会显示最新的一条。再往下则会优先推荐他们公司的相关内容，比如百度文库等。但这些并不是想要的信息。而校园网搜索引擎主要对象是学校师生，通过分析我们自己实现的搜索引擎可以发现，搜索结果都是跟学校、关键字主题相关的内容，而且结果会随着关键字的命中率的减少而依次排序，这位我们获取信息提供了相对准确的途径。

## 5.3 测试总结

通过测试，可以证明本系统的原型系统相对于普通搜索引擎来说能够更准确、可靠的提供校内新闻，对最新的校园网中的信息及时收录，从而保证用户获取信息的准确性和及时性。目前系统针对“常熟理工学院新闻网”消息，能够很好的进行收录、索引、检索，结果相对满意。但由于校园内网的复杂度，校园网下又有很多二级域名，下属机构等，爬虫还需对这些页面进行深度的爬取，这些内容等待以后系统完善会继续实现。

# 结束语

本文实现的“常熟理工学院校园网搜索引擎”原型系统的设计与实现初步成功。相对与目前的校园网搜索新闻内容只能以标题形式进行搜索，这套系统大大提高了搜索的准确度。

本系统采用Python语言实现，采用Scrapy和Django基于Python的开源框架，结合ElasticSearch搜索引擎实现了校园网搜索引擎的设计与实现。总的来说，这套系统的设计难度不是很大，对于搜索引擎来说最难的应该是爬虫爬取部分，搜索引擎应该满足信息的完整性和实时性，所以爬取的内容要新、要全。由于常熟理工学院校园网并没有采取任何反爬虫的措施，所以对于信息的爬取来说并不难。ElasticSearch和Django实现的交互部分也没有太多的问题。

由于之前没有接触过Python，所有东西现学现用，所以很多细节在实现上没有考虑完整。比如，在常熟理工主页下分设了许多机构，由于这些机构的页面采用的开发语言、技术、页面细节等方面都不一样所以对于爬取来说增加了很大的难度。所以目前来说，该原型系统是以“常熟理工学院新闻网”进行分析与设计，所涉及内容均为该站点下的内容资源。对于其他页面内容，待以后根据具体需求具体添加。

通过对本系统的设计与开发，掌握了不少知识的使用。也加深了自己的开发经验。该系统经过测试，结果达到初步于其，能够投入使用。该系统在一定意义上具有一定的参考性和实用性，至于系统其他内容，根据以后的具体需求在具体进行开发完善。

# 参考文献与附录

1. 搜索引擎优化(搜索优化) - 搜狗百科.[Z] http://baike.sogou.com/h234029.htm?sp=l167574169
2. 白晓丹. 搜索引擎网页相关性及检索效率评价体系研究[D].北京交通大学,2015.
3. 杨成宝. 我国搜索引擎市场发展趋势与策略研究[D].山东大学,2011.
4. 刘鸣. 面向电子商务的垂直搜索引擎的研究和实现[D].云南师范大学,2016.
5. 安子建. 基于Scrapy框架的网络爬虫实现与数据抓取分析[D].吉林大学,2017.
6. 邓铮. 基于网页分块思想的搜索引擎索引系统[D].天津大学,2009
7. 米硕,孙瑞彬,李欣,明晓,赵汝程.Scrapy分布式爬虫原理分析与概述[J].中国新通信,2018,20(04):234.
8. 李乐思. 学生公寓房源数据采集平台的设计与实现[D].北京交通大学,2017.
9. 张宗华,屈英,叶志佳,牛新征.基于多特征匹配和Bloom filter的重复数据删除算法[J].深圳大学学报(理工版),2016,33(05):531-535.
10. 刘石磊.对反爬虫网站的应对策略[J].电脑知识与技术,2017,13(15):19-21+23.
11. 曾亚飞. 基于Elasticsearch的分布式智能搜索引擎的研究与实现[D].重庆大学,2016.
12. Django官方网站. [Z]. https://www.djangoproject.com/
13. Python基础教程.[Z] http://www.runoob.com/python/python-tutorial.html
14. Scrapy官方文档. Scrapy文档首页 [OL].

http://scrapy-chs.readthedocs.io/zh\_CN/0.24/

1. [1]米硕,孙瑞彬,李欣,明晓,赵汝程.Scrapy分布式爬虫原理分析与概述[J].中国新通信,2018,20(04):234.

# 致谢

首先我要感谢我的指导老师，计算机科学与工程学院学院的沈建老师。在系统分析设计前，沈建老师提供了较多的技术选型供参考，也提供了很多的学习文档，使我在开发过程中少走了很多弯路。在系统设计过程中，沈建老师积极指导、热心解答各个问题，给我的系统设计和完成论文提供了非常大帮助。

然后，还要感谢我工作单位的领导、同事在我系统设计与论文编写过程中提供的帮助。在工作时，领导关心毕设的完成情况，同事也在技术上提供了不少的帮助。人事部门的同事也在最后给我几天的假期回来安心写论文。

此外，我的朋友们、同事们也给我的毕设带来了巨大的帮助，给我带来极大的启发，再次表示诚挚的感谢。也要感谢此次论文参考文献中的作者们，他们的文章给我了极大的启发和指导作用，少走了很多弯路。

感谢论文评阅老师们在百忙之中对我的论文进行审阅。

最后，对以上提到的各位朋友、同事、老师表示真诚的谢意。