第 5 章

视频讲解

爬虫应用——校园网搜索引擎

5.1 校园网搜索引擎功能分析

随着校园网建设的迅速发展，校园网内的信息内容正在以惊人的速度增加着。如何更全面、更准确地获取最新、最有效的信息已经成为人们把握机遇、迎接挑战和获取成功的重要条件。目前虽然已经有了像 Google、百度这样优秀的通用搜索引擎，但是它们并不能适用于所有的情况和需要。对学术搜索、校园网的搜索来说，一个合理的排序结果是非常重要的。另外，互联网上的信息量巨大，远远超出哪怕是最大的一个搜索引擎可以完全收集的能力范围。本章旨在使用 Python 建立一个适合校园网使用的 Web 搜索引擎系统，它能在较短的时间内爬取页面信息，具有有效、准确的中文分词功能，能实现对校园网上新闻信息的快速检索展示。

5.2 校园网搜索引擎系统设计

视频讲解

校园网搜索引擎一般需要以下几个步骤。

(1) 网络爬虫爬取这个网站，得到所有网页链接。

网络爬虫就是一只会嗅着 URL(链接)爬过成千上万个网页，并把网页内容搬到用户计算机上供用户使用的苦力虫子。如图 5-1 所示，给定爬虫的出发页面 A 的 URL，它就从起始页 A 出发，读取 A 的所有内容，并从中找到 5 个 URL，分别指向页面 B、C、D、E 和 F，然后它顺着链接依次抓取 B、C、D、E 和 F 页面的内容，并从中发现新的链接，再沿着链接爬到新的页面，对爬虫带回来的网页内容分析链接，继续爬到新的页面，以此类推，直到找不到新的链接或者满足了人为设定的停止条件为止。

至于这只虫子前进的方式，则分为广度优先搜索(BFS)和深度优先搜索(DFS)。在这张图中 BFS 的搜索顺序是 A-B-C-D-E-F-G-H-I，而深度优先搜索的顺序是遍历的路径，即

A-F-G　E-H-I　B C D。

(2) 得到网页的源代码,解析剥离出想要的新闻内容、标题、作者等信息。

(3) 把所有网页的新闻内容做成词条索引,一般采用倒排表索引。

索引一般有正排索引(正向索引)和倒排索引(反向索引)两种类型。

① 正排索引(正向索引,forward index):正排表是以文档的 ID 为关键字,表中记录文档(即网页)中每个字或词的位置信息,查找时扫描表中每个文档中字或词的信息直到找出所有包含查询关键字的文档。

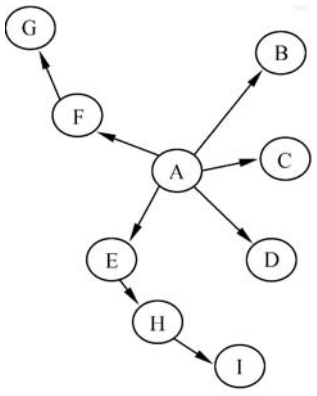

图 5-1　网站链接示意图

正排表的结构如图 5-2 所示,这种组织方法在建立索引的时候结构比较简单,建立比较方便且易于维护;因为索引是基于文档建立的,若是有新的文档加入,直接为该文档建立一个新的索引块,挂接在原来索引文件的后面。若是有文档删除,则直接找到该文档号文档对应的索引信息,将其直接删除。但是在查询的时候需要对所有的文档进行扫描以确保没有遗漏,这样就使得检索时间大大延长,检索效率低下。

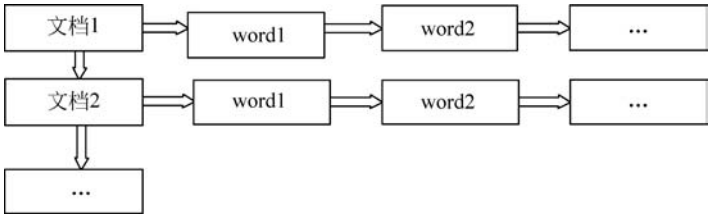

图 5-2　正排表结构示意图

尽管正排表的工作原理非常简单,但是由于其检索效率太低,除非在特定情况下,否则实用价值不大。

② 倒排索引(反向索引,inverted index):倒排表以字或词为关键字进行索引,表中关键字所对应的记录表项记录了出现这个字或词的所有文档,一个表项就是一个字表段,它记录该文档的 ID 和字符在该文档中出现的位置情况。

由于每个字或词对应的文档数量在动态变化,所以倒排表的建立和维护都较为复杂,但是在查询的时候由于可以一次得到查询关键字所对应的所有文档,所以效率高于正排表。在全文检索中,检索的快速响应是一个最为关键的性能,而索引的建立由于在后台进行,尽管效率相对低一些,但不会影响整个搜索引擎的效率。

倒排表的结构如图 5-3 所示。

正排索引是从文档到关键字的映射(已知文档求关键字),倒排索引是从关键字到文档的映射(已知关键字求文档)。

在搜索引擎中每个文件都对应一个文件 ID,文件内容被表示为一系列关键词的集合(实际上,在搜索引擎索引库中关键词已经转换为关键词 ID)。例如“文档 1”经过分词提取了 20 个关键词,每个关键词都会记录它在文档中的出现次数和出现位置,得到正向索引的结构如下:

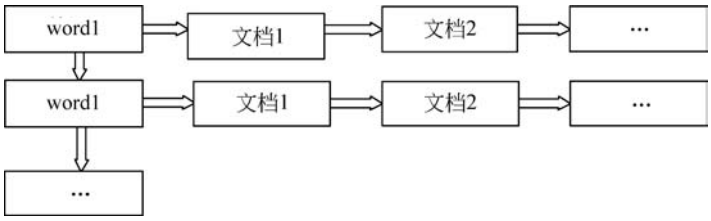

图 5-3　倒排表结构示意图

“文档 1”的 ID > 单词 1:出现次数,出现位置列表;单词 2:出现次数,出现位置列表;…
“文档 2”的 ID > 此文档出现的关键词列表

当用户搜索关键词“华为手机”时,假设只存在正向索引(forward index),那么就需要扫描索引库中的所有文档,找出所有包含关键词“华为手机”的文档,再根据打分模型进行打分,排出名次后呈现给用户。因为互联网上收录在搜索引擎中的文档的数目是个天文数字,这样的索引结构根本无法满足实时返回排名结果的要求。所以搜索引擎会将正向索引重新构建为倒排索引,即把文件 ID 对应到关键词的映射转换为关键词到文件 ID 的映射,每个关键词都对应一系列的文件,这些文件中都出现这个关键词,得到倒排索引的结构如下:

“单词 1”:“文档 1”的 ID,“文档 2”的 ID,…
“单词 2”:带有此关键词的文档 ID 列表

(4) 搜索时,根据搜索词在词条索引中查询,按顺序返回相关的搜索结果,也可以按网页评价排名顺序返回相关的搜索结果。

当用户输入一串搜索字符串时,程序会先进行分词,然后依照每个词的索引找到相应网页。假如在搜索框中输入“从前有座山山里有座庙小和尚”,搜索引擎首先会对字符串进行分词处理“从前/有/座山/山里/有/座庙/小和尚”,然后按照一定的规则对词做布尔运算,例如每个词之间做“与”运算,在索引中搜索“同时”包含这些词的页面。

所以本系统主要由以下 4 个模块组成。

- 信息采集模块:主要是利用网络爬虫实现对校园网信息的抓取。
- 索引模块:负责对爬取的新闻网页的标题、内容和作者进行分词并建立倒排词表。
- 网页排名模块:TF/IDF 是一种统计方法,用于评估一字词对于一个文件集或一个语料库中的一份文件的重要程度。
- 用户搜索界面模块:负责用户关键字的输入以及搜索结果信息的返回。

5.3　关键技术

5.3.1　正则表达式


视频讲解

把网页中的超链接提取出来需要使用正则表达式。那么什么是正则表达式?在回答这个问题之前先来看一看为什么要有正则表达式。

在编程处理文本的过程中,经常需要按照某种规则去查找一些特定的字符串。例如知道一个网页上的图片都叫“image/8554278135.jpg”之类的名字,只是那串数字不一样;又或者在一堆人员的电子档案中,要把他们的电话号码全部找出来,整理成通讯录。诸如此类工作,可不可以利用这些规律,让程序自动来做这些事情?答案是肯定的。这时候就需要一种描述这些规律的方法,正则表达式就是描述文本规则的代码。

正则表达式是一种用来匹配字符串文本的强有力的武器。它是用一种描述性的语言来给字符串定义一个规则。凡是符合规则的字符串,就认为它“匹配”了,否则该字符串就是不合法的。

❶ 正则表达式的语法

正则表达式并不是 Python 中特有的功能,它是一种通用的方法,要使用它必须会用正则表达式来描述文本规则。

正则表达式使用特殊的语法来表示,表 5-1 列出了正则表达式的语法。

表 5-1　正则表达式的语法

模　式	描　述
^	匹配字符串的开头
$	匹配字符串的末尾
.	匹配任意字符,除了换行符
[...]	用来表示一组字符,例如[amk]匹配'a'、'm'或'k';[0-9]匹配任何数字,类似于[0123456789];[a-z]匹配任何小写字母;[a-zA-Z0-9]匹配任何字母及数字
[^...]	不在[]中的字符,例如[^abc]匹配除了 a、b、c 之外的字符;[^0-9]匹配除了数字之外的字符
*	数量词,匹配 0 个或多个
+	数量词,匹配 1 个或多个
?	数量词,以非贪婪方式匹配 0 个或 1 个
{ n,}	重复 n 次或更多次
{ n, m}	重复 n～m 次
a|b	匹配 a 或 b
(re)	匹配括号内的表达式,也表示一个组
(?imx)	正则表达式包含 3 种可选标志,即 i、m、x,只影响括号中的区域
(?-imx)	正则表达式关闭 i、m 或 x 可选标志,只影响括号中的区域
(?: re)	类似(...),但是不表示一个组
(?imx: re)	在括号中使用 i、m 或 x 可选标志
(?-imx: re)	在括号中不使用 i、m 或 x 可选标志
(?= re)	前向肯定界定符,如果所含正则表达式以…表示,在当前位置成功匹配时成功,否则失败。一旦所含表达式已经尝试,匹配引擎根本没有提高,模式的剩余部分还要尝试界定符的右边
(?!re)	前向否定界定符,与前面肯定界定符相反,当所含表达式不能在字符串当前位置匹配时成功
(?> re)	匹配的独立模式,省去回溯
\w	匹配字母、数字及下画线,等价于'[A-Za-z0-9_]'
\W	匹配非字母、数字及下画线,等价于 '[^A-Za-z0-9_]'

续表

模　式	描　述
\s	匹配任何空白字符，包括空格、制表符、换页符等，等价于[\f\n\r\t\v]
\S	匹配任何非空白字符，等价于[^\f\n\r\t\v]
\d	匹配任意数字，等价于[0-9]
\D	匹配任意非数字，等价于[^0-9]
\A	匹配字符串开始
\Z	匹配字符串结束，如果存在换行，只匹配到换行前的结束字符串
\z	匹配字符串结束
\G	匹配最后匹配完成的位置
\b	匹配一个单词边界，也就是单词和空格间的位置。例如，'er\b'可以匹配"never"中的'er'，但不能匹配"verb"中的'er'
\B	匹配非单词边界，例如'er\B'能匹配"verb"中的'er'，但不能匹配"never"中的'er'
\n、\t 等	匹配一个换行符、一个制表符等

正则表达式通常用于在文本中查找匹配的字符串。在 Python 中数量词默认是贪婪的，总是尝试匹配尽可能多的字符；非贪婪的则相反，总是尝试匹配尽可能少的字符。例如，正则表达式"ab * "如果用于查找"abbbc"，将找到"abbb"；如果使用非贪婪的数量词"ab * ?"，将找到"a"。

在正则表达式中，如果直接给出字符，就是精确匹配。从正则表达式语法中能够了解到用\d 可以匹配一个数字，用\w 可以匹配一个字母或数字，用. 可以匹配任意字符，所以模式'00\d'可以匹配'007'，但无法匹配'00A'；模式'\d\d\d'可以匹配'010'；模式'\w\w\d'可以匹配'py3'；模式'py. '可以匹配'pyc'、'pyo'、'py!'，等等。

如果要匹配变长的字符，在正则表达式模式字符串中用 * 表示任意个字符(包括 0 个)，用＋表示至少一个字符，用?表示 0 个或 1 个字符，用{n}表示 n 个字符，用{n,m}表示 n～m 个字符。这里看一个复杂的表示电话号码的例子，即\d{3}\s＋\d{3,8}。

从左到右解读一下：

\d{3}表示匹配 3 个数字，例如'010'；

\s 可以匹配一个空格(也包括 Tab 等空白符)，所以\s＋表示至少有一个空格；

\d{3,8}表示 3～8 个数字，例如'67665230'。

综合起来，上面的正则表达式可以匹配以任意个空格隔开的带区号的电话号码。

如果要匹配'010-67665230'这样的号码，怎么办？由于'-'是特殊字符，在正则表达式中要用'\'转义，所以上面的正则表达式是\d{3}\-\d{3,8}。

如果要做更精确的匹配，可以用[]表示范围，例如：

[0-9a-zA-Z_]可以匹配一个数字、字母或者下画线；

[0-9a-zA-Z_]＋可以匹配至少由一个数字、字母或者下画线组成的字符串，例如'a100'、'0_Z'、'Py3000'等；

[a-zA-Z_][0-9a-zA-Z_] * 可以匹配以字母或下画线开头，后接任意个由一个数字、字母或者下画线组成的字符串，也就是 Python 合法的变量；

[a-zA-Z_][0-9a-zA-Z_]{0，19}更精确地限制了变量的长度是 1～20 个字符(前面 1 个字符＋后面最多 19 个字符)。

另外,A|B 可以匹配 A 或 B,所以(P|p)ython 可以匹配'Python'或者'python'。

^表示行的开头,^\d 表示必须以数字开头。

$ 表示行的结束,\d$ 表示必须以数字结束。

❷ re 模块

Python 提供了 re 模块,包含所有正则表达式的功能。

1) match()方法

re.match()的格式为 re.match(pattern, string, flags)。

其第 1 个参数是正则表达式,第 2 个参数表示要匹配的字符串;第 3 个参数是标志位,用于控制正则表达式的匹配方式,例如是否区分大小写、多行匹配等。

match()方法判断是否匹配,如果匹配成功,返回一个 Match 对象,否则返回 None。常见的判断方法如下:

```
test = '用户输入的字符串'
if re.match(r'正则表达式', test):          #r 前缀为原义字符串,它表示对字符串不进行转义
    print('ok')
else:
    print('failed')
```

例如:

```
>>> import re
>>> re.match(r'^\d{3}\-\d{3,8}$', '010-12345')       #返回一个 Match 对象
<_sre.SRE_Match object; span=(0,9), match='010-12345'>
>>> re.match(r'^\d{3}\-\d{3,8}$', '010 12345')       #'010 12345'不匹配规则,返回 None
```

Match 对象是一次匹配的结果,包含了很多关于此次匹配的信息,可以使用 Match 提供的可读属性或方法来获取这些信息。

Match 属性如下。

- string:匹配时使用的文本。
- re:匹配时使用的 Pattern 对象。
- pos:文本中正则表达式开始搜索的索引,值与 Pattern.match()和 Pattern.search()方法的同名参数相同。
- endpos:文本中正则表达式结束搜索的索引,值与 Pattern.match()和 Pattern.search()方法的同名参数相同。
- lastindex:最后一个被捕获的分组在文本中的索引。如果没有被捕获的分组,将为 None。
- lastgroup:最后一个被捕获的分组的别名。如果这个分组没有别名或者没有被捕获的分组,将为 None。

Match 方法如下:

- group([group1, …]):获得一个或多个分组截获的字符串,当指定多个参数时将以元组形式返回。参数 group1 可以使用编号也可以使用别名;编号 0 代表整个匹配的子串;当不填写参数时返回 group(0);没有截获字符串的组返回 None;截获了多次的组返回最后一次截获的子串。

- groups([default])：以元组形式返回全部分组截获的字符串，相当于调用 group(1，2，…，last)。default 表示没有截获字符串的组以这个值替代，默认为 None。
- groupdict([default])：返回已有别名的组的别名为键、以该组截获的子串为值的字典，没有别名的组不包含在内。default 的含义同上。
- start([group])：返回指定的组截获的子串在 string 中的起始索引(子串第 1 个字符的索引)。group 的默认值为 0。
- end([group])：返回指定的组截获的子串在 string 中的结束索引(子串最后一个字符的索引+1)。group 的默认值为 0。
- span([group])：返回(start(group)，end(group))。

Match 对象的相关属性和方法示例如下：

```
import re
t = "19:05:25"
m = re.match(r'^(\d\d)\:(\d\d)\:(\d\d)$', t)      #r 原义
print("m.string:", m.string)            #m.string: 19:05:25
print(m.re)                             #re.compile('^(\\d\\d)\\:(\\d\\d)\\:((\\d\\d))$')
print("m.pos:", m.pos)                  #m.pos: 0
print("m.endpos:", m.endpos)            #m.endpos: 8
print("m.lastindex:", m.lastindex)      #m.lastindex: 3
print("m.lastgroup:", m.lastgroup)      #m.lastgroup: None
print("m.group(0):", m.group(0))        #m.group(0): 19:05:25
print("m.group(1,2):", m.group(1, 2))   #m.group(1,2):('19', '05')
print("m.groups():", m.groups())        #m.groups():('19', '05', '25')
print("m.groupdict():", m.groupdict()) #m.groupdict(): {}
print("m.start(2):", m.start(2))        #m.start(2): 3
print("m.end(2):", m.end(2))            #m.end(2): 5
print("m.span(2):", m.span(2))          #m.span(2):(3, 5)
```

关于分组的内容见后面。

2) 分组

除了简单地判断是否匹配之外，正则表达式还有提取子串的强大功能，用()表示的就是要提取的分组。例如^(\d{3})-(\d{3,8})$ 分别定义了两个组，可以直接从匹配的字符串中提取出区号和本地号码：

```
>>> m = re.match(r'^(\d{3})-(\d{3,8})$', '010-12345')
>>> m.group(0)       #'010-12345'
>>> m.group(1)       #'010'
>>> m.group(2)       #'12345'
```

如果正则表达式中定义了组，就可以在 Match 对象上用 group()方法提取出子串。注意 group(0)永远是原始字符串，group(1)、group(2)表示第 1、2 个子串。

3) 切分字符串

用正则表达式切分字符串比用固定的字符更灵活，请看普通字符串的切分代码：

```
>>> 'a b   c'.split(' ')                #split(' ')表示按空格分隔
['a', 'b', '', '', 'c']
```

其结果是无法识别连续的空格,可以使用 re.split()方法来分隔字符串,例如 re.split(r'\s+', text)将字符串按空格分隔成一个单词列表:

```
>>> re.split(r'\s+', 'a b   c')                    #用正则表达式
['a', 'b', 'c']
```

无论多少个空格都可以正常分隔。

再如分隔符既有空格又有逗号、分号的情况:

```
>>> re.split(r'[\s\,]+', 'a,b, c  d')          #可以识别空格、逗号
['a', 'b', 'c', 'd']
>>> re.split(r'[\s\,\;]+', 'a,b;; c  d')       #可以识别空格、逗号、分号
['a', 'b', 'c', 'd']
```

4) search()和 findall()方法

re.match()总是从字符串"开头"去匹配,并返回匹配的字符串的 Match 对象,所以当用 re.match()去匹配非"开头"部分的字符串时会返回 None。

```
str1 = 'Hello World!'
print(re.match(r'World',str1))              #结果为 None
```

如果想在字符串中的任意位置去匹配,使用 re.search()或 re.findall()。

re.search()将对整个字符串进行搜索,并返回第 1 个匹配的字符串的 Match 对象。

```
str1 = 'Hello World!'
print(re.search(r'World',str1))
```

输出结果如下:

```
<_sre.SRE_Match object; span=(6,11), match='World'>
```

re.findall()函数将返回一个所有匹配的字符串的字符串列表。例如:

```
str1 = 'Hi, I am Shirley Hilton. I am his wife.'
>>> print(re.search(r'hi',str1))
```

以上代码的输出结果如下:

```
<_sre.SRE_Match object; span=(10, 12), match='hi'>
```

此时应用 re.findall()函数:

```
>>> re.findall(r'hi',str1)
```

输出结果如下:

```
['hi', 'hi']
```

这两个"hi"分别来自"Shirley"和"his"。在默认情况下正则表达式是严格区分大小写的,所以"Hi"和"Hilton"中的"Hi"被忽略了。

如果只想找到"hi"这个单词,而不把包含它的单词计算在内,那就可以使用"\bhi\b"这个正则表达式。"\b"在正则表达式中表示单词的开头或结尾,空格、标点、换行都算是单词

的分隔；而“\b”自身又不会匹配任何字符，它代表的只是一个位置，所以单词前后的空格、标点之类不会出现在结果中。

在前面的例子中，“\bhi\b”匹配不到任何结果，因为没有单词 hi（“Hi”不是，严格区分大小写）。但如果是“\bhi”，就可以匹配到 1 个“hi”，出自“his”。

5.3.2 中文分词

在英文中，单词之间是以空格作为自然分界符的；而中文只是句子和段可以通过明显的分界符来简单划分，唯独词没有一个形式上的分界符，虽然也同样存在短语之间的划分问题，但是在词这一层上，中文要比英文复杂得多。

中文分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。中文分词是网页分析索引的基础。分词的准确性对搜索引擎来说十分重要，如果分词速度太慢，即使再准确，对于搜索引擎来说也是不可用的，因为搜索引擎需要处理很多网页，如果分析消耗的时间过长，会严重影响搜索引擎内容更新的速度。因此，搜索引擎对于分词的准确率和速率都提出了很高的要求。

jieba 是一个支持中文分词、高准确率、高效率的 Python 中文分词组件，它支持繁体分词和自定义词典，并支持 3 种分词模式。

（1）精确模式：试图将句子最精确地切开，适合文本分析。

（2）全模式：把句子中所有可以成词的词语都扫描出来，速度非常快，但是不能解决歧义的问题。

（3）搜索引擎模式：在精确模式的基础上对长词再次切分，提高召回率，适合用于搜索引擎分词。

5.3.3 安装和使用 jieba

在命令行中输入以下代码：

```
pip install  jieba
```

如果出现以下提示则安装成功。

```
Installing collected packages: jieba
  Running setup.py install for jieba ... done
Successfully installed jieba-0.38
```

组件提供了 jieba.cut()方法用于分词，cut()方法接受两个输入参数：

（1）第 1 个参数为需要分词的字符串。

（2）cut_all 参数用来控制分词模式。

jieba.cut()返回的结构是一个可迭代的生成器(generator)，可以使用 for 循环来获得分词后得到的每个词语，也可以用 list(jieba.cut(...))转化为 list 列表。例如：

```
import jieba
seg_list = jieba.cut("我来到清华大学", cut_all = True)      # 全模式
```

```
print("Full Mode:", '/'.join(seg_list))
seg_list = jieba.cut("我来到清华大学")                    #默认是精确模式,或者 cut_all = False
print(type(seg_list))                                    #<class 'generator'>
print("Default Mode:", '/'.join(seg_list))
seg_list = jieba.cut_for_search("我来到清华大学")          #搜索引擎模式
print("搜索引擎模式:", '/'.join(seg_list))
seg_list = jieba.cut("我来到清华大学")
for word in seg_list:
    print(word,end = ' ')
```

运行结果如下：

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\ADMINI~1\AppData\Local\Temp\jieba.cache
Loading model cost 1.648 seconds.
Prefix dict has been built successfully.
Full Mode: 我/来到/清华/清华大学/华大/大学
<class 'generator'>
Default Mode: 我/来到/清华大学
搜索引擎模式: 我/来到/清华/华大/大学/清华大学
我 来到 清华大学
```

jieba.cut_for_search()方法仅有一个参数，为分词的字符串，该方法适用于搜索引擎构造倒排索引的分词，粒度比较细。

5.3.4 为 jieba 添加自定义词典

国家 5A 级景区存在很多与旅游相关的专有名词，举个例子：

[输入文本] 故宫的著名景点包括乾清宫、太和殿和黄琉璃瓦等

[精确模式] 故宫/的/著名景点/包括/乾/清宫/、/太和殿/和/黄/琉璃瓦/等

[全模式]故宫/的/著名/著名景点/景点/包括/乾/清宫/太和/太和殿/和/黄/琉璃/琉璃瓦/等

显然，专有名词乾清宫、太和殿、黄琉璃瓦(假设为一个文物)可能因分词而分开，这也是很多分词工具的一个缺陷。但是 jieba 分词支持开发者使用自定义的词典，以便包含 jieba 词库中没有的词语。虽然 jieba 有新词识别能力，但自行添加新词可以保证更高的正确率，尤其是专有名词。

其基本用法如下：

```
jieba.load_userdict(file_name)                    #file_name 为自定义词典的路径
```

词典格式是一个词占一行；每行分 3 个部分，一部分为词语，另一部分为词频，最后为词性(可省略，jieba 的词性标注方式和 ICTCLAS 的标注方式一样。ns 为地点名词，nz 为其他专用名词，a 是形容词，v 是动词，d 是副词)，3 个部分用空格隔开。例如以下自定义词典 dict.txt：

```
乾清宫 5 ns
黄琉璃瓦 4
```

```
云计算 5
李小福 2 nr
八一双鹿 3 nz
凯特琳 2 nz
```

下面是导入自定义词典后再分词。

```
import jieba
jieba.load_userdict("dict.txt")                    #导入自定义词典
text = "故宫的著名景点包括乾清宫、太和殿和黄琉璃瓦等"
seg_list = jieba.cut(text, cut_all = False)        #精确模式
print("[精确模式]: ", "/ ".join(seg_list))
```

输出结果如下，其中专有名词连在一起，即乾清宫和黄琉璃瓦。

```
[精确模式]:故宫/ 的/ 著名景点/ 包括/ 乾清宫/ 、/ 太和殿/ 和/ 黄琉璃瓦/ 等
```

5.3.5 文本分类的关键词提取

当文本分类时，在构建 VSM(向量空间模型)的过程中或者把文本转换成数学形式的计算中，需要运用到关键词提取技术，jieba 可以简便地提取关键词。

其基本用法如下：

```
jieba.analyse.extract_tags(sentence, topK = 20, withWeight  = False, allowPOS = ())
```

这里需要先 import jieba.analyse。其中，sentence 为待提取的文本；topK 为返回几个 TF/IDF 权重最大的关键词，默认值为 20；withWeight 为是否一并返回关键词权重值，默认值为 False；allowPOS 指仅包含指定词性的词，默认值为空，即不进行筛选。

```
import jieba,jieba.analyse
jieba.load_userdict("dict.txt")                                    #导入自定义词典
text = "故宫的著名景点包括乾清宫、太和殿和午门等.其中乾清宫非常精美,午门是紫禁城的正门,
午门居中向阳."
seg_list = jieba.cut(text, cut_all = False)
print("分词结果:", "/".join(seg_list))                             #精确模式
tags = jieba.analyse.extract_tags(text, topK = 5)                  #获取关键词
print("关键词:", " ".join(tags))
tags = jieba.analyse.extract_tags(text, topK = 5,withWeight = True)
                                                                   #返回关键词权重值
print(tags)
```

输出结果如下：

```
分词结果: 故宫/的/著名景点/包括/乾清宫/、/太和殿/和/午门/等/./其中/乾清宫/非常/精
美/,/午门/是/紫禁城/的/正门/,/午门/居中/向阳/.
关键词: 午门 乾清宫 著名景点 太和殿 向阳
[('午门', 1.5925323525975001),('乾清宫', 1.4943459378625),('著名景点',   0.86879235325),
('太和殿', 0.63518800210625),('向阳', 0.578517922051875)]
```

其中,午门出现 3 次、乾清宫出现 2 次、著名景点出现 1 次。如果 topK=5,按照顺序输出提取 5 个关键词,则输出“午门 乾清宫 著名景点 太和殿 向阳”。

```
jieba.analyse.TFIDF(idf_path = None)                    #新建 TF/IDF 实例,idf_path 为 IDF 频率文件
```

关键词提取所使用的逆向文件频率(IDF)文本语料库可以切换成自定义语料库的路径。

```
jieba.analyse.set_idf_path(file_name)                   #file_name 为自定义语料库的路径
```

关键词提取所使用的停止词(Stop Words)文本语料库可以切换成自定义语料库的路径。

说明:TF/IDF 是一种统计方法,用于评估一字词对于一个文件集或一个语料库中的一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加,但同时会随着它在语料库中出现的频率成反比下降。TF/IDF 的主要思想是:如果某个词或短语在一篇文章中出现的频率 TF 高,并且在其他文章中很少出现,则认为此词或者短语具有很好的类别区分能力,适合用来分类。

5.3.6 deque

deque(double-ended queue 的缩写)双向队列类似于 list 列表,位于 Python 标准库 collections 中。它提供了两端都可以操作的序列,这意味着在序列的前后都可以执行添加或删除操作。

❶ 创建双向队列 deque

```
from collections import deque
d = deque()
```

❷ 添加元素

```
d = deque()
d.append(3)
d.append(8)
d.append(1)
```

此时 d=deque([3,8,1])、len(d)=3、d[0]=3、d[-1]=1。

deque 支持从任意一端添加元素。append()用于从右端添加一个元素,appendleft()用于从左端添加一个元素。

❸ 两端都使用 pop

```
d = deque(['1', '2', '3', '4', '5'])
```

d. pop()抛出的是'5',d. popleft()抛出的是'1',可见默认 pop()抛出的是最后一个元素。

❹ 限制 deque 的长度

```
d = deque(maxlen = 20)
for i in range(30):
    d.append(str(i))
```

此时 d 的值为 d=deque(['10', '11', '12', '13', '14', '15', '16', '17', '18', '19','20', '21', '22', '23', '24', '25', '26', '27', '28', '29'], maxlen=20),可见当限制长度的 deque 增加超过限制数的项时,另一边的项会自动删除。

❺ 添加 list 的各项到 deque 中

```
d = deque([1,2,3,4,5])
d.extend([0])
```

此时 d=deque([1,2,3,4,5,0])。

```
d. extendleft([6,7,8])
```

此时 d=deque([8,7,6,1,2,3,4,5,0])。

5.4　程序设计的步骤

5.4.1　信息采集模块——网络爬虫的实现


视频讲解

网络爬虫的实现原理及过程如下:

(1) 获取初始的 URL。初始的 URL 地址可以由用户指定的某个或某几个初始爬取网页决定。

(2) 根据初始的 URL 爬取页面并获得新的 URL。在获得初始的 URL 地址之后,首先需要爬取对应 URL 地址中的网页,在爬取了对应的 URL 地址中的网页后将网页存储到原始数据库中,并且在爬取网页的同时发现新的 URL 地址,将已爬取的 URL 地址存放到一个已爬取 URL 列表中,用于去重复判断爬取的进程。

(3) 将新的 URL 放到 URL 队列中。注意,在第(2)步中获取了下一个新的 URL 地址之后会将新的 URL 地址放到 URL 队列中。

(4) 从 URL 队列中读取新的 URL,并依据新的 URL 爬取网页,同时从新网页中获取新 URL,并重复上述的爬取过程。

(5) 当满足爬虫系统设置的停止条件时停止爬取。在编写爬虫的时候一般会设置相应的停止条件,如果没有设置停止条件,爬虫会一直爬取下去,直到无法获取新的 URL 地址为止,若设置了停止条件,爬虫则会在停止条件满足时停止爬取。

根据图 5-4 所示的网络爬虫的实现原理及过程,这里指定中原工学院新闻门户的 URL 地址"http://www.zut.edu.cn/index/xwdt.htm"为初始的 URL。

使用 unvisited 队列存储待爬取 URL 链接的集合并使用广度优先搜索,使用 visited 集合存储已访问过的 URL 链接。

```
unvisited = deque()                    #待爬取链接的列表,使用广度优先搜索
visited = set()                        #已访问的链接集合
```

在数据库中建立两个 table,其中一个是 doc 表,存储每个网页 ID 和 URL 链接。

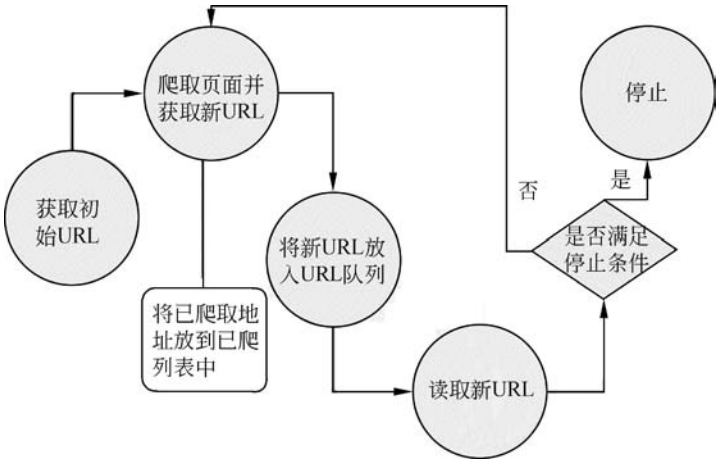

图 5-4　网络爬虫的实现原理及过程

```
create table doc(id int primary key,link text)
```

例如：

```
1   http://www.zut.edu.cn/index/xwdt.htm
2   http://www.zut.edu.cn/info/1052/19838.htm
3   http://www.zut.edu.cn/info/1052/19837.htm
4   http://www.zut.edu.cn/info/1052/19836.htm
5   http://www.zut.edu.cn/info/1052/19835.htm
6   http://www.zut.edu.cn/info/1052/19834.htm
7   http://www.zut.edu.cn/info/1052/19833.htm
…
```

另一个是 word 表，即倒排表，存储词语和其对应的网页 ID 的 list。

```
create table word(term varchar(25) primary key,list text)
```

如果一个词在某个网页中出现多次，那么 list 中这个网页的序号也出现多次。list 最后转换成一个字符串存进数据库。

例如，词“王宗敏”出现在网页 ID 为 12、35、88 号的网页里，12 号页面 1 次，35 号页面 3 次，88 号页面 2 次，它的 list 应为[12,35,35,35,88,88]，转换成字符串"12 35 35 35 88 88"存储在 word 表的一条记录中，形式如下：

term	list
王宗敏	12　35　35　35　88　88
校友会	54　190　190　701　986　986　1024

爬取中原工学院新闻网页的代码如下：

```
#search_engine_build-2.py
import sys
from collections import deque
```

```
import urllib
from urllib import request
import re
from bs4 import BeautifulSoup
import lxml
import sqlite3
import jieba

url = 'http://www.zut.edu.cn'          #入口

unvisited = deque()                    #待爬取链接的集合,使用广度优先搜索
visited = set()                        #已访问的链接集合
unvisited.append(url)

conn = sqlite3.connect('viewsdu.db')
c = conn.cursor()
#在创建表之前先删除表是因为之前测试的时候已经建过 table 了,所以再次运行代码
#的时候要把旧的 table 删了重建
c.execute('drop table doc')
c.execute('create table doc(id int primary key,link text)')
c.execute('drop table word')
c.execute('create table word(term varchar(25) primary key,list text)')
conn.commit()
conn.close()
print('**************** 开始爬取 ************************** ')
cnt = 0
print('开始..... ')
while unvisited:
    url = unvisited.popleft()
    visited.add(url)
    cnt += 1
    print('开始抓取第',cnt,'个链接:',url)

    #爬取网页内容
    try:
        response = request.urlopen(url)
        content = response.read().decode('utf-8')

    except:
        continue
    #寻找下一个可爬取的链接,因为搜索范围是网站内,所以对链接有格式要求,需根据具体情况
    #而定解析网页内容,可能有几种情况,这也是根据这个网站网页的具体情况写的
    soup = BeautifulSoup(content,'lxml')
    all_a = soup.find_all('a',{'class':"c67214"})     #本页面所有的新闻链接<a>
    for a in all_a:
        #print(a.attrs['href'])
        x = a.attrs['href']            #网址
        if re.match(r'http.+',x):   #排除是 http 开头,而不是 http://www.zut.edu.cn 网址
            if not re.match(r'http\:\/\/www\.zut\.edu\.cn\/.+',x):
                continue
```

```
            if re.match(r'\/info\/. + ',x):                    #"/info/1046/20314.htm"
                x = 'http://www.zut.edu.cn' + x
            elif re.match(r'info/. + ',x):                     #"info/1046/20314.htm"
                x = 'http://www.zut.edu.cn/' + x
            elif re.match(r'\.\.\/info/. + ',x):               #"../info/1046/20314.htm"
                x = 'http://www.zut.edu.cn' + x[2:]
            elif re.match(r'\.\.\/\.\.\/info/. + ',x):         #"../../info/1046/20314.htm"
                x = 'http://www.zut.edu.cn' + x[5:]
            #print(x)
            if(x not in visited) and(x not in unvisited):
                    unvisited.append(x)

        a = soup.find('a',{'class':"Next"})                    #下一页<a>
        if a!= None:
            x = a.attrs['href']                                #网址
            if re.match(r'xwdt\/. + ',x):
                x = 'http://www.zut.edu.cn/index/' + x
            else:
                x = 'http://www.zut.edu.cn/index/xwdt/' + x
            if(x not in visited) and(x not in unvisited):
               unvisited.append(x)
```

以上代码实现要爬取的网址队列 unvisited。

5.4.2 索引模块——建立倒排词表

解析新闻网页内容,这个过程需要根据这个网站网页的具体情况来处理。

```
soup = BeautifulSoup(content,'lxml')
title = soup.title
article = soup.find('div',class_ = 'c67215_content',id = 'vsb_newscontent')
author = soup.find('span',class_ = "authorstyle67215")      #作者
time = soup.find('span',class_ = "timestyle67215")
if title == None and article == None and author == None:
    print('无内容的页面.')
    continue
elif article == None and author == None:
    print('只有标题.')
    title = title.text
    title = ''.join(title.split())
    article = ''
    author = ''
elif article == None:
    print('有标题有作者,缺失内容')
    title = title.text
    title = ''.join(title.split())
    article = ''
    author = author.get_text("",strip = True)
    author = ''.join(author.split())
```

```
    elif author == None:
        print('有标题有内容,缺失作者')
        title = title.text
        title = ''.join(title.split())
        article = article.get_text("",strip = True)
        article = ''.join(article.split())
        author = ''
    else:
        title = title.text
        title = ''.join(title.split())
        article = article.get_text("",strip = True)
        article = ''.join(article.split())
        author = author.get_text("",strip = True)
        author = ''.join(author.split())
    print('网页标题:',title)
```

提取的网页内容存在于 title、article、author 中,对它们进行中文分词,并对每个分出的词语建立倒排词表。

```
        seggen = jieba.cut_for_search(title)
        seglist = list(seggen)
        seggen = jieba.cut_for_search(article)
        seglist += list(seggen)
        seggen = jieba.cut_for_search(author)
        seglist += list(seggen)

        #数据存储
        conn = sqlite3.connect("viewsdu.db")
        c = conn.cursor()
        c.execute('insert into doc values(?,?)',(cnt,url))
        #对每个分出的词语建立倒排词表
        for word in seglist:
            #print(word)
            #检验看看这个词语是否已存在于数据库
            c.execute('select list from word where term = ?',(word,))
            result = c.fetchall()
            #如果不存在
            if len(result) == 0:
                docliststr = str(cnt)
                c.execute('insert into word values(?,?)',(word,docliststr))
            #如果已存在
            else:
                docliststr = result[0][0]          #得到字符串
                docliststr += ' ' + str(cnt)
                c.execute('update word set list = ? where term = ?',(docliststr,word))
        conn.commit()
        conn.close()
    print('词表建立完毕!!')
```

以上代码只需运行一次即可,搜索引擎所需的数据库已经建好。运行上述代码出现如下结果:

```
开始抓取第 110 个链接: http://www.zut.edu.cn/info/1041/20191.htm
网页标题: 我校 2017 年学生奖助项目评审工作完成资助育人成效显著 - 中原工学院
开始抓取第 111 个链接: http://www.zut.edu.cn/info/1041/20190.htm
网页标题: 我校教师李慕杰、王学鹏参加中国致公党河南省第一次代表大会 - 中原工学院
开始抓取第 112 个链接: http://www.zut.edu.cn/info/1041/20187.htm
网页标题: 我校与励展企业开展校企合作 - 中原工学院
开始抓取第 113 个链接: http://www.zut.edu.cn/info/1041/20184.htm
网页标题: 平顶山学院李培副校长一行来我校考察交流 - 中原工学院
开始抓取第 114 个链接: http://www.zut.edu.cn/info/1041/20179.htm
网页标题: 我校学生在工程造价技能大赛中获佳绩 - 中原工学院
开始抓取第 115 个链接: http://www.zut.edu.cn/info/1041/20178.htm
网页标题: 我校召开 2018 届毕业生就业工作会议 - 中原工学院
```


视频讲解

5.4.3 网页排名和搜索模块

当需要搜索的时候执行 search_engine_use.py,完成网页排名和搜索功能。

网页排名采用 TF/IDF 统计。TF/IDF 是一种用于信息检索与数据挖掘的常用加权技术。TF/IDF 统计用于评估一词对于一个文件集或一个语料库中的一份文件的重要程度。TF 的意思是词频(Term Frequency),IDF 的意思是逆文本频率指数(Inverse Document Frequency)。TF 表示词条 t 在文档 d 中出现的频率。IDF 的主要思想是:如果包含词条 t 的文档越少,则词条 t 的 IDF 越大,说明词条 t 具有很好的类别区分能力。

词条 t 的 IDF 计算公式如下:

$$idf = \log(N/df)$$

其中,N 是文档总数,df 是包含词条 t 的文档数量。

在本程序中 tf={文档号:出现次数}存储的是某个词在文档中出现的次数。例如王宗敏的 tf={12:1,35:3,88:2}即词“王宗敏”出现在网页 ID 为 12、35、88 号的网页里,12 号页面 1 次,35 号页面 3 次,88 号页面 2 次。

score={文档号:文档得分}用于存储命中(搜到)文档的排名得分。

```
#search_engine_use.py
import re
import urllib
from urllib import request
from collections import deque
from bs4 import BeautifulSoup
import lxml
import sqlite3
import jieba
import math
conn = sqlite3.connect("viewsdu.db")
c = conn.cursor()
c.execute('select count( * ) from doc')
```

```
N = 1 + c.fetchall()[0][0]                          #文档总数
target = input('请输入搜索词:')
seggen = jieba.cut_for_search(target)               #将搜索内容分词
score = {}                                          #字典,用于存储"文档号:文档得分"
for word in seggen:
    print('得到查询词:',word)
    tf = {}                                         #文档号:次数{12:1,35:3,88:2}
    c.execute('select list from word where term = ?',(word,))
    result = c.fetchall()
    if len(result)> 0:
        doclist = result[0][0]                      #字符串"12 35 35 35 88 88"
        doclist = doclist.split(' ')
        doclist = [int(x) for x in doclist]         #['12','35','35','35','88','88']
                                    #把字符串转换成元素为 int 的 list[12,35,88]
        df = len(set(doclist))      #当前 word 对应的 df 数,注意 set 集合实现去掉重复项
        idf = math.log(N/df)        #计算出 IDF
        print('idf:',idf)
        for num in doclist:         #计算词频 TF,即在某文档中出现的次数
            if num in tf:
                tf[num] = tf[num] + 1
            else:
                tf[num] = 1
        #TF 统计结束,现在开始计算 score
        for num in tf:
            if num in score:
                #如果该 num 文档已经有分数了,则累加
                score[num] = score[num] + tf[num] * idf
            else:
                score[num] = tf[num] * idf
sortedlist = sorted(score.items(),key = lambda d:d[1],reverse = True)
                                                    #对 score 字典按字典的值排序
#print('得分列表',sortedlist)
cnt = 0
for num,docscore in sortedlist:
    cnt = cnt + 1
    c.execute('select link from doc where id = ?',(num,))
                                                    #按照 ID 获取文档的连接(网址)
    url = c.fetchall()[0][0]
    print(url ,'得分:',docscore)                     #输出网址和对应得分
    try:
        response = request.urlopen(url)
        content = response.read().decode('utf-8')   #可以输出网页内容
    except:
        print('oops...读取网页出错')
        continue
    #解析网页输出标题
    soup = BeautifulSoup(content,'lxml')
    title = soup.title
    if title == None:
        print('No title.')
```

```
        else:
            title = title.text
            print(title)
        if cnt > 20:                                    # 超过 20 条则结束,即输出前 20 条
            break
    if cnt == 0:
        print('无搜索结果')
```

当运行 search_engine_use.py 时出现如下提示：

```
请输入搜索词:王宗敏
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\xmj\AppData\Local\Temp\jieba.cache
Loading model cost 0.961 seconds.
Prefix dict has been built successfully.
得到查询词:王宗敏
idf:3.337509562404897
http://www.zut.edu.cn/info/1041/20120.htm 得分:13.350038249619589
王宗敏校长一行参加深圳校友会年会并走访合作企业 - 中原工学院
http://www.zut.edu.cn/info/1041/20435.htm 得分:13.350038249619589
中国工程院张彦仲院士莅临我校指导工作 - 中原工学院
http://www.zut.edu.cn/info/1041/19775.htm 得分:10.012528687214692
我校河南省功能性纺织材料重点实验室接受现场评估 - 中原工学院
http://www.zut.edu.cn/info/1041/19756.htm 得分:10.012528687214692
王宗敏校长召开会议推进"十三五"规划"八项工程"建设 - 中原工学院
http://www.zut.edu.cn/info/1041/19726.htm 得分:10.012528687214692
我校 2017 级新生开学典礼隆重举行 - 中原工学院
```

说明：由于中原工学院网站 2022 年 4 月改版，程序代码访问的原网站已改为 www1.zut.edu.cn，所以代码中出现的 www.zut.edu.cn 都修改为 www1.zut.edu.cn 即可正常运行。