

一、题型

简答：3×5'

仅供参考↓

综合题：4x20'+1x15'

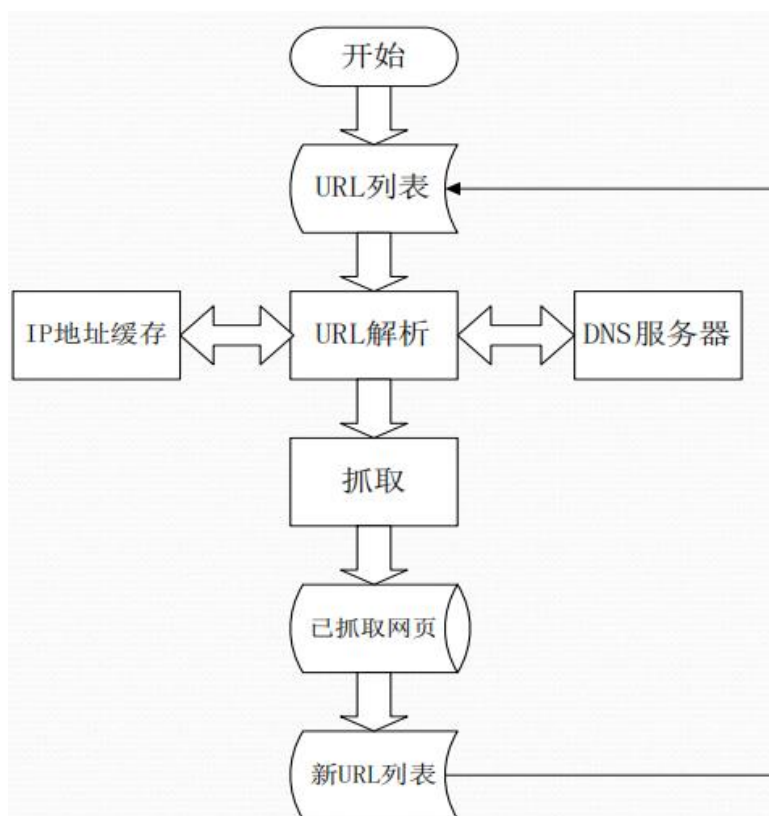
二、Chapt3 网页抓取技术

2.1 爬虫概念

网络爬虫（Crawler），也称为蜘蛛程序（Spider），或网络机器人（Robots）。网络爬虫是一个自动提取网页的程序，尽可能多和快的给索引部分输送网页。通过网页的链接地址来寻找网页，从网站某一个页面开始，读取网页的内容，找到在网页中的其它链接地址，然后通过这些链接地址寻找下一个网页，一直循环下去，直到把这个网站所有的网页都抓取完为止。

2.2 爬虫工作原理/基本流程

图例（助记）



原理/流程

在爬虫开始的时候，需要给爬虫输送一个URL列表，列表中的URL地址就是爬虫的起始位置。爬虫从这些URL出发，开始爬行，一直不断地发现新的URL，然后再根据策略爬行这些新发现的URL，如此永远反复下去。一般的爬虫都自己建立DNS缓冲，建立

DNS缓冲的目的是加快URL解析成IP地址的速度。

2.3 广/深度优先策略

深度优先策略是尽量往最远的地方走，直到不能再走为止；使用该策略容易爬行很多重复的结点，需要能控制爬行路径、避免路径重复的算法。

广度优先策略是一种层次型距离不断增大的遍历方法。广度优先更适合爬虫的分布式处理，启动多个爬虫，每个爬虫负责一层。此外，重要的网页往往离种子站点距离较近，随着不断的浏览，所看到的网页的重要性越来越低。

2.4 Robots协议是什么

2.4.1 协议内容

Robots协议是Web站点和搜索引擎爬虫交互的一种方式，Robots.txt文本文件规定爬虫只抓取指定内容，或禁止爬虫抓取网站的某些内容。当一个爬虫访问一个站点时，它会首先检查该站点根目录下是否存在robots.txt，如果存在，爬虫就会按照该文件中的内容来确定访问的范围；不存在，爬虫就沿着链接抓取。

2.4.2 使用格式

- (1) User-agent:
 - 用于描述搜索引擎爬虫的名字，在“Robots.txt”文件中，如果有多条User-agent记录说明有多个搜索引擎爬虫会受到该协议的限制，对该文件来说，至少要有一条User-agent记录。如果该项的值设为*，则该协议对任何搜索引擎爬虫均有效。
- (2) Disallow:
 - 用于描述不希望被访问到的一个URL，这个URL可以是一条完整的路径，也可以是部分的，任何以Disallow开头的URL均不会被Robot访问到。

2.4.3 例题

例题 1

- 通过“/robots.txt”禁止所有搜索引擎爬虫抓取“/bin/cgi/”目录，以及“/tmp/”目录和/foo.html文件，设置方法如下：
 - User-agent: *
 - Disallow: /bin/cgi/
 - Disallow: /tmp/
 - Disallow: /foo.html

例题 2

- 通过 “/robots.txt”只允许某个搜索引擎抓取，而禁止其他的搜索引擎抓取。如：只允许名为 “slurp”的搜索引擎爬虫抓取，而拒绝其他的搜索引擎爬虫抓取 “/cgi/”目录下的内容，设置方法如下：
- User-agent: *
- Disallow: /cgi/
- User-agent: slurp
- Disallow:

清华大学出版社

例题 3

- 禁止任何搜索引擎抓取我的网站，方法如下：
- User-agent: *
- Disallow: /
- 只禁止某个搜索引擎抓取我的网站。如：只禁止名为 “slurp”的搜索引擎蜘蛛抓取，方法如下：
- User-agent: slurp
- Disallow: /

例题 4

(2) 利用Robots协议禁止所有搜索引擎爬虫爬取/bin/目录/禁止所有搜索引擎爬虫爬取/xyz/目录下的foo.html,禁止所有搜索引擎爬虫爬取/temp/目录下后缀名为.html的文件,请写出实现上述功能的Robots协议内容。(4分)

User-agent: *
Disallow: /bin/
Disallow: /xyz/foo.html
Disallow: /temp/*.html

三、Chapt4 网页爬虫开发技术

3.1 正则表达式

3.1.1 概念

正则表达式是一个特殊的字符序列，用于检查一个字符串是否与某种模式匹配。是描述字符串排列的一套规则。

是用于处理字符串的工具，拥有独特的语法以及一个独立的处理引擎，功能强大。

3.1.2 语法

1. 行定位符

行定位符是用来描述字符串的边界。“^”表示行开始，“\$”表示行结尾，如“^de”表示以 de 开头的字符串，“de\$”表示以 de 结尾的字符串。

2. 单词定界符

在查找一个单词的时候，如 ar 是否在一个字符串“How are you”中存在，很明显，如果匹配的话，ar 肯定是串“How are you”中的一部分。怎样才能让其匹配单词，而不是单词的一部分呢？这时候，可以使用一个单词定界符\b。例如：

用\b ar\b 去匹配“How are you”，就会提示匹配不到。

另外一个单词定界符\B，它的意思和\b 正好相反，它匹配的字符串不能是一个完整的单词，而是其他单词或字符串中的一部分。例如：

用\B ar\B 去匹配“How are you”，就会匹配到。

3. 选择字符

有两个选择字符“[]”与“|”，表示或。例如，ab|ba 表示 ab 或者 ba。使用“[]”只能匹配单个字符，而“|”可以匹配任意长度的字符串。在使用“[]”的时候，往往配合连接字符“-”一起使用，例如：

[a-d]表示 a 或 b 或 c 或 d

eg: [a-zA-Z0-9]+

表示所有(字母以及数字)

排除字符“^”，表示排除操作。正则表达式提供了“^”来表示排除不符合的字符，^一

般放在[]中。例如：

[^1-5]，表示不是 1~5 的数字。

5. 限定符

限定符主要是用来限定每个字符串出现的次数。常用的限定符如表 4-1 所示。

表 4-1 常用限定符

限定符	说 明	举 例
?	匹配前面的字符零次或一次 eg: <.+?> 匹配 <.+>	colou? r, 该表达式可以匹配 colour 和 color
+	匹配前面的字符一次或多次	go+gle, 该表达式可以匹配的范围为 gogle~goo...gel
*	匹配前面的字符零次或多次	go*gle, 该表达式可以匹配的范围为 gggle~goo...gel
{n}	匹配前面的字符 n 次	go{2}gle, 该表达式只匹配 google
{n,}	匹配前面的字符最少 n 次	go{2,}gle, 该表达式可以匹配的范围为 google~goo...gel
{n,m}	匹配前面的字符最少 n 次, 最多 m 次	employe{0,2}, 该表达式可以匹配 employ, employe 和 employee

6. 元字符

正则表达式里面有很多元字符,常用的元字符如表 4-2 所示。

表 4-2 常用元字符

元字符	说 明	举 例
.	匹配除换行符以外的任意字符	匹配 mr\nM\tR 中的 m,r,M,\t,R
\w	匹配字母、数字、下划线或汉字 <i>\w+ 匹配所有字母、数字、下划线、汉字...</i>	匹配“howareyou\n”中的 h,o,w,a,r,e,y,o,u,但不能匹配\n
\W	匹配除字母、数字、下划线或汉字以外的字符 <i>\W+</i>	匹配“howareyou\n”中的\n,但不能匹配 h,o,w,a,r,e,y,o,u
\s	匹配单个的空白符(包括 Tab 和换行符) <i>\s+</i>	匹配 mr\tMR 中的\t
\S	匹配除单个空白符以外的所有字符 <i>\S+</i>	匹配 mr\tMR 中的 m,r,M,R
\d	匹配数字 <i>\d+ [0-9]</i>	可以与 how7M 中的数字 7 匹配
\D	匹配非数字字符 <i>\D+</i>	

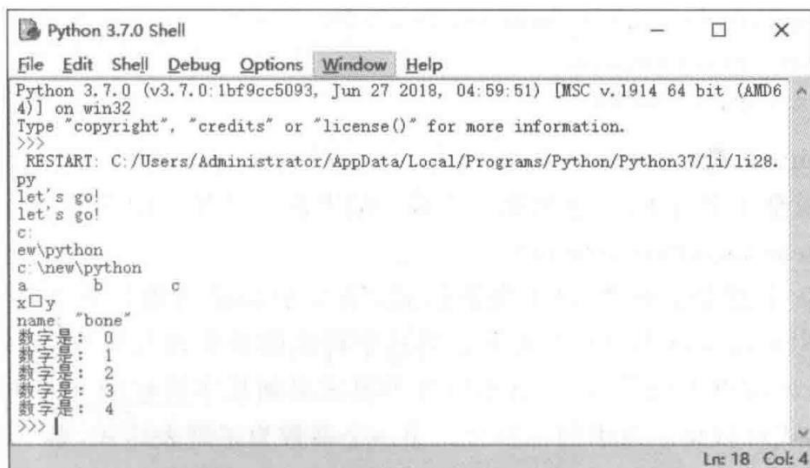
7. 转义字符

转义字符“\”主要是将一些特殊字符转为普通字符。而这些常用特殊字符有“.”“?”“\”等。下面举例说明转义字符的使用。

\. 代表. 而不是元字符.

```
a = 'let\'s go!'           # 转义单引号
print(a)
b = "let's go!"           # 字符串内是双引号,就用单引号来区分开
print(b)
c = "c:\new\python"       # 换行
print(c)
d = "c:\\new\\python"
print(d)
print("a\tb\tc")          # 使用制表符
print("x\ay")             # 执行时系统会有提示音
print("name: \"bone\"")   # 转义双引号
for i in range(5):         # \r 表示回车
    print("\r 数字是:", i)
```

运行结果如图 4-12 所示。



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/Administrator/AppData/Local/Programs/Python/Python37/li/li28.
py
let's go!
let's go!
c:
ew\python
c:\new\python
a      b      c
x□y
name: "bone"
数字是: 0
数字是: 1
数字是: 2
数字是: 3
数字是: 4
>>> |
```

图 4-12 转义字符的使用

3.1.3 四个模块函数

3.1.4 例题

3.2 爬虫框架（了解）

Scrapy	爬虫框架
Crawley	爬虫框架
PySpider	爬虫框架
Portia	爬虫框架

四、Chapt5 网页信息预处理

4.1 DOM树（了解）

4.1.1 概念

DOM即文档对象模型，是一种将文档（如HTML、XML等）表示为树形结构的对象模型，提供了一种结构化的方式来访问和操作文档中的内容和元素。

4.1.2 DOM树建立过程

- ①建立标签分析栈。
- ②顺序读取网页标签并依次入栈。
- ③文本结点不入栈。
- ④成对标签同时退栈。

DOM树建立以后，遍历树中的每个结点，将其中的文本送到分词模块进行处理。

4.2 文本处理

4.2.1 列举三种中文分词方法

- ①基于词典的分词方法（或“基于字符串匹配的分词方法”），也叫机械分词方法。
- ②基于统计的分词方法。
- ③基于理解的分词方法。

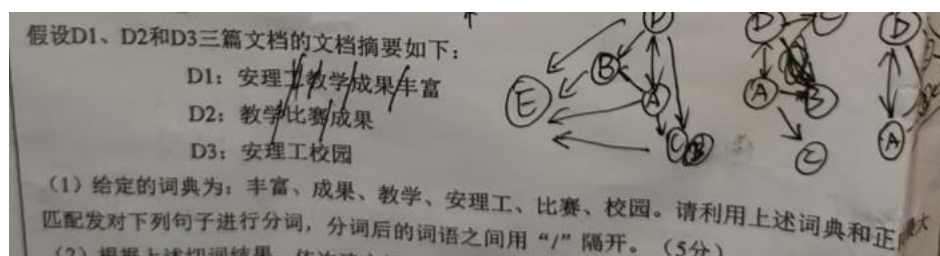
4.2.2 详述基于词典的分词方法的原理

①正向最大匹配法(FMM)

算法思想

匹配方向是从左向右。选取包含 6~8 个汉字的符号串作为最大符号串，把最大符号串与词典中的单词条目相匹配，如果不能匹配，就减掉一个汉字继续匹配，直到在词典中找到相应的单词为止。

例题



①：安理工 / 教学 / 成果 / 丰富

②：教学 / 比赛 / 成果

③：安理工 / 校园

②逆向最大匹配法 (BMM)

算法思想

匹配方向是从左到右。与正向最大匹配法类似。

例题

①（安理工教学成果丰富）：丰富 / 成果 / 教学 / 安理工

②（教学比赛成果）：成果 / 比赛 / 教学

③（安理工校园）：校园 / 安理工

五、Chapt6 信息索引技术

5.1 顺排检索

5.1.1 规则

最终需要画的表格

地址	检索词	条件满足指向	条件不满足指向	级位
1	A
2	B
...	...	命中	落选	...

填写依据

(1) 填写

地址：从A到字母X_n依次为 1, 2, ..., n。

检索词：按表达式从左到右的字母，依次填入。

级位：X、(X) 的级位为 0；(X 遇左括号的级位加 1；X) 遇右括号的级位减 1；
括号内的其他情况，级位不变。

条件满足指向 (AFD)：X*，针对*运算符，填入指向后一词的地址。

条件不满足指向 (NFD)：X+，针对+运算符，填入指向后一词的地址。

Ps：最后一个检索词的AFD、NFD必为命中、落选。

(2) 补全

针对AFD和NFD内容，从下往上补全，设地址为a_i。

AFD

若a_i的级位 > a_(i+1)的级位，则将a_(i+1)的AFD内容复制到a_i的AFD内容；

否则比较a_i与a_(i+2)...不断向下比较，直至找到可复制的内容为止。“命中、落选”也是可复制的内容。

NFD

若a_i的级位 ≥ a_(i+1)的级位，则将a_(i+1)的AFD内容复制到a_i的AFD内容；

否则比较a_i与a_(i+2)...不断向下比较，直至找到可复制的内容为止。“命中、落选”也是可复制的内容。

(3) 观察有无负号

若有，则在-X的所在地址行，对其“条件满足指向”和“条件不满足指向”的内容添加一个“相反”的符号即可。即：

...  ...

5.1.2 例题

例题 1

$A * B + C$ PS: 2³ → 角标代表填写顺序

地址	检索词	条件满足指向	条件不满足指向	级位
1	A	2 ³	3 ⁶	0
2	B	命中 ⁵	3 ⁴	0
3	C	命中 ¹	落选 ²	0

例题 2

$A * (B + C) + (-D) * (E + F)$

地址	检索词	条件满足指向	条件不满足指向	级位
1	A	2 ³	4 ¹²	0
2	B	命中 ¹¹	3 ⁴	1
3	C	命中 ¹⁰	4 ⁵	0
4	D	5 ⁶	落选 ⁹	0
5	E	命中 ⁸	6 ⁷	1
6	F	命中 ¹	落选 ²	0

例题 3

$(A * B + C) * (-D + E)$

地址	检索词	条件满足指向	条件不满足指向	级位
1	A	2	3	1
2	B	4	3	1
3	C	4	落选	0
4	D	命中	5	1
5	E	命中	落选	0

5.2 倒排索引

5.2.1 例题 1

Doc 1:
清华 / 大学 / 清华 / 主页
Doc 2: 世纪 / 清华
Doc 3: 北京 / 大学

根据上述切词结果,依次建立这三篇文档的词项列表和正排索引表。根据正排索引表,给出从正排索引表到倒排索引表的建立过程。查询: 清华大学。

(1) 词项列表:

WordID	Term	DF
0	清华	2
1	大学	2
2	主页	1
3	世纪	1
4	北京	1

正排索引表:

Doc ID	Word ID	No. of Hit	Hit List
Doc1	0	2	0000 0002
	1	1	0001
	2	1	0003
Doc2	3	1	0000
	0	1	0001
Doc3	4	1	0000
	1	1	0001

(2) ①补全 DocId. ②按 word ID 排序

DocId	WordID	No. of Hit	Hit List
Doc1	0	2	0000 0002
Doc1	1	1	0001
Doc1	2	1	0003
Doc2	3	1	0000
Doc2	0	1	0001
Doc3	4	1	0000
Doc3	1	1	0001

(3) ③交换 wordID 与 DocID

WordID	DocID	No. of Hit	Hit List
0	Doc1	2	0000 0002
0	Doc2	1	0001
1	Doc1	1	0001
1	Doc3	1	0001
2	Doc1	1	0003
3	Doc2	1	0000
4	Doc3	1	0000

(4) ④整合 wordID, 生成倒排索引表.

WordID	Term	DF	Pointer
0	清华	2	
1	大学	2	
2	主页	1	
3	世纪	1	
4	北京	1	

(5) ⑤检索倒排索引表

WordID	DocID	No. of Hit	Hit List
0	Doc1	2	0000 0002
1	Doc2	1	0001
1	Doc1	1	0001
1	Doc3	1	0001
2	Doc1	1	0003
3	Doc2	1	0000
4	Doc3	1	0000

(6) ⑥清华/大学

WordID	Term	DF	Pointer
0	清华	2	
1	大学	2	
2	主页	1	
3	世纪	1	
4	北京	1	

(7) ⑦得到词项表

WordID	DocID	No. of Hit	Hit List
0			
1			

(8) ⑧查询结果: WordID DocID List

指向③中的 0, 1, 2, 3, 4

5.2.2 例题 2

D①: 安理工 / 教学 / 成果 / 丰富

D②: 教学 / 比赛 / 成果

D③: 安理工 / 校园

根据上述切词结果,依次建立这三篇文档的词项列表和正排索引表。根据正排索引表,给出从正排索引表到倒排索引表的建立过程。查询: 比赛成果。

(1) 词项表

WordID	Term	DF
0	安理	2
1	教学	2
2	成果	2
3	丰富	1
4	比赛	1
5	校园	1

正排索引表:

DocID	WordID	No.ofHit	HitList
D1	0	1	0000
	1	1	0001
	2	1	0002
	3	1	0003
D2	1	1	0000
	4	1	0001
	2	1	0002
D3	0	1	0000
	5	1	0001

(2) ① 补全DocID

DocID	WordID	No.ofHit	HitList
D1	0	1	0000
D1	1	1	0001
D1	2	1	0002
D1	3	1	0003
D2	1	1	0000
D2	4	1	0001
D2	2	1	0002
D3	0	1	0000
D3	5	1	0001

② 按WordID排序

DocID	WordID	No.ofHit	HitList
	0		
	0		
	1		
	1		
	2		
	2		(略)
	3		
	4		
	5		

③ 交换DocID与WordID

WordID	DocID	No.ofHit	HitList
0			
0			
1			
1	(略)		
2			
2			
3			
4			
5			

④ 合WordID, 生成倒排索引表

WordID	Term	DF	Pointer	WordID	DocID	No.ofHit	HitList
0			→ 0				
1			→ 1				
2	(略)		→ 2				(略)
3			→ 0				
4			→ 3				
5			→ 4				
			→ 5				

(3) ① 比赛/成果

② 检查词项表

WordID	Term	DF	Pointer
0			
1			
2			
3			
4			
5			

③ 检查倒排索引表

WordID	DocID	No.ofHit	HitList
0			
1			
2			
3			
4			
5			

④ 索引表压缩

WordID	DocID	No.ofHit	HitList
2			
4			

⑤ 结果查询:

WordID	DocID	List
4, 2	D2	

注意顺序

5.3 逆波兰表达式

5.3.1 规则

Ps: 后缀表达式也叫“逆波兰表达式”。

在计算机中，中缀表达式转后缀表达式时需要借助一个栈，用于保存暂时还不能确定运算顺序的运算符。从左到右依次扫描中缀表达式中的每一项，具体转化过程如下：计算机方法

1) 遇到操作数。直接加入后缀表达式。
 2) 遇到界限符。若为“（”，则直接入栈；若为“）”，则不入栈，且依次弹出栈中的运算符（op），并加入后缀表达式，直到遇到“（”为止，并直接删除“（”。
 3) 遇到运算符。① 若其优先级高于栈顶运算符或遇到栈顶为“（”，则直接入栈；② 若其优先级低于或等于栈顶运算符，则依次弹出栈中的运算符并加入后缀表达式，直到遇到一个优先级低于它的运算符或遇到“（”或栈空为止，之后将当前运算符入栈。
 4) 按上述方法扫描所有字符后，将栈中剩余运算符依次弹出，并加入后缀表达式。

Handwritten notes:
 - 用于存放暂时还不能确定的运算符 (op)
 - 或栈为空
 - 优先级
 - 栈顶

5.3.2 例题

写表顺序: 1, 2, 3, 4, 5

解: ① 序号	② 表达式	目标数据⑤	堆栈④	说明③
1	$a \times b + c \times (d + e)$ #		#	#表示结束符
2	$a \times b + c \times (d + e)$ #	a	#	a写入内存
3	$a \times b + c \times (d + e)$ #	a	#*	* > #
4	$a \times b + c \times (d + e)$ #	ab	#*	b写入内存
5	$a \times b + c \times (d + e)$ #	ab*	#	+ < *
6	$a \times b + c \times (d + e)$ #	ab*	#+	+ > #
7	$a \times b + c \times (d + e)$ #	ab*c	#+	c写入内存
8	$a \times b + c \times (d + e)$ #	ab*c	#+*	* > +
9	$a \times b + c \times (d + e)$ #	ab*c	#+*(左括号“(”入栈
10	$a \times b + c \times (d + e)$ #	ab*cd	#+*(d写入内存
11	$a \times b + c \times (d + e)$ #	ab*cd	#+*(+	+ > (
12	$a \times b + c \times (d + e)$ #	ab*cde	#+*(+	e写入内存
13	$a \times b + c \times (d + e)$ #	ab*cde+	#+*	右括号“)”遇左括号“(”出栈
14	$a \times b + c \times (d + e)$ #	ab*cde+*	#	#出栈

请使用黑色字迹书写，在本题中区域内作答，超出黑色矩形边框限定区域的答案无效

5.4 文本压缩技术

5.4.1 哈夫曼编码

规则

- ①统计频率。将原始符号按照出现概率递减(或递增)的顺序排列;
- ②将两个最小出现概率进行合并相加, 得到的结果作为新符号的出现概率;
- ③重复进行步骤①和②直到概率相加的结果等于1为止;
- ④分配码字。将形成的二叉树左结点标0, 右结点标1 (或左结点标1, 右结点标0), 从根结点回溯到原始符号, 记录根结点到当前符号之间的0, 1序列, 从而得到每个符号的编码。
- 因为每个编码都是通过树上从根开始的不同路径得到的, 所以没有一个编码是其他编码的前缀。

Ps: 哈夫曼树不唯一。

例题

在整个索引文档库中索引词出现的概率如下表。利用霍夫曼树对索引词进行编码, 画出霍夫曼树。根据霍夫曼编码, 进行编码分配, 写出最后的编码分配表。

索引词	丰富	成果	教学	安理工	比赛	校园
出现的概率	0.05	0.15	0.2	0.35	0.1	0.15

解析

(1)

(2)

字符	权	编码
丰富	0.05	0000
成果	0.15	001
教学	0.2	011
安理工	0.35	1
比赛	0.1	0001
校园	0.15	010

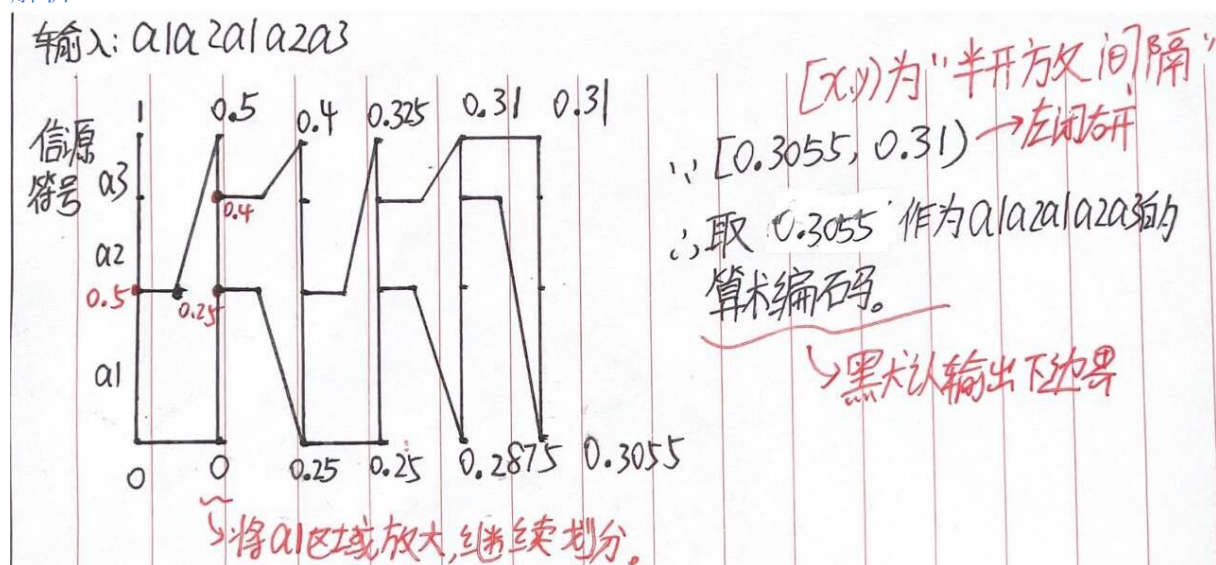
5.4.2 算术编码

例题 1

下表是信源符号的概率，求符号序列 $a_1a_2a_1a_2a_3$ 的算术编码。

符号	a_1	a_2	a_3
概率	0.5	0.3	0.2

解析 1

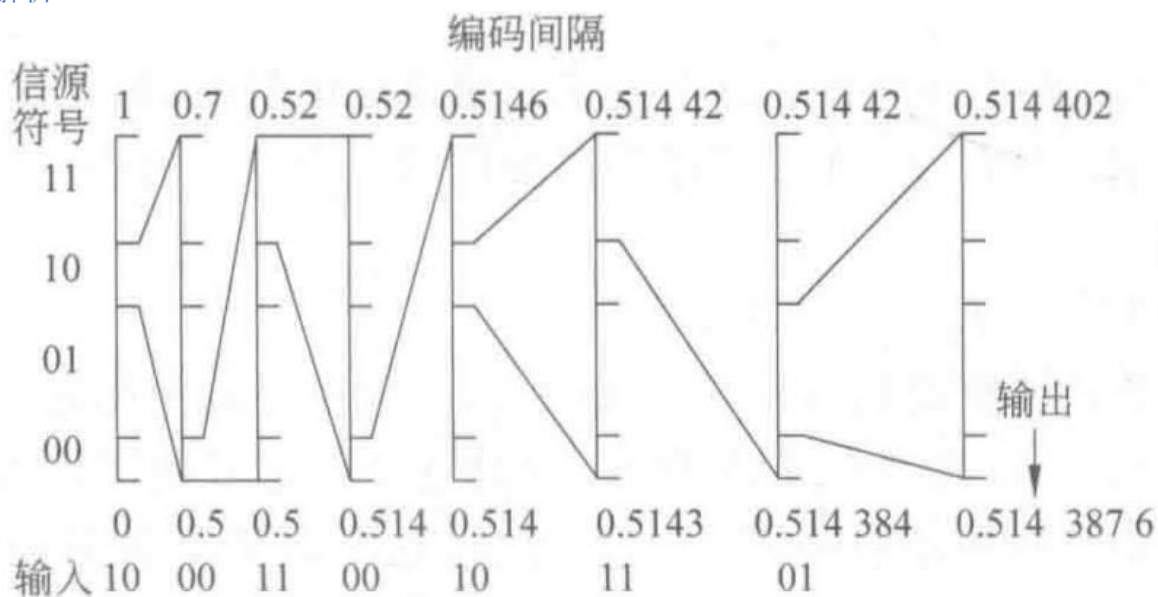


例题 2

假设信源符号为 $\{00, 01, 10, 11\}$ ，这些符号的概率分别为 $\{0.1, 0.4, 0.2, 0.3\}$ ，

求 10 00 11 00 10 11 01 的算术编码。

解析 2



六、Chapt7 信息查询与评价技术

6.1 向量空间模型

6.1.1 内积相似度

内积相似度越大，相关度越高。

已知: $Q = (x_1, y_1, z_1)$, $D = (x_2, y_2, z_2)$

内积相似度: $\text{Sim}(Q, D) = x_1x_2 + y_1y_2 + z_1z_2$

6.1.2 余弦相似度

余弦相似度越接近 1，相关度越高。

余弦相似度: $\text{Sim}(Q, D) = \cos\alpha = \frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{(x_1^2 + y_1^2 + z_1^2)} \sqrt{(x_2^2 + y_2^2 + z_2^2)}}$

$$= \frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{x_1^2 + y_1^2 + z_1^2} \cdot \sqrt{x_2^2 + y_2^2 + z_2^2}}$$

6.1.3 例题

查询向量 $Q = (0.6, 0.8, 0)$ ，三个文档 D_1 、 D_2 和 D_3 的文档向量为 $D_1 = (0.3, 0.6, 0.2)$ 、 $D_2 = (0.6, 0.3, 0.3)$ 和 $D_3 = (0.6, 0.1, 0.3)$ 。(1) 请分别计算查询向量与三个文档的内积相似度，并结合该相似度说明哪个文档与查询相关度更高？(2) 请分别计算查询向量与三个文档的余弦相似度，并结合该相似度说明哪个文档与查询相关度更高？

内积相似度

(1) $\text{Sim}(Q, D_1) = 0.6 \times 0.3 + 0.8 \times 0.6 + 0 = 0.66$ 内积相似度越大，相关度越高。

$\text{Sim}(Q, D_2) = 0.6 \times 0.6 + 0.8 \times 0.3 + 0 = 0.6$ 且 $0.66 > 0.6 > 0.44$

$\text{Sim}(Q, D_3) = 0.6 \times 0.6 + 0.8 \times 0.1 + 0 = 0.44$ 文档 D_1 与查询向量 Q 的相关度更高。

(2) 余弦相似度

$\cos\alpha = \text{Sim}(Q, D_1) = \frac{0.66}{\sqrt{(0.6^2 + 0.8^2)} \times \sqrt{(0.3^2 + 0.6^2 + 0.2^2)}} = \frac{0.66}{0.7} \approx 0.943$

$\cos\beta = \text{Sim}(Q, D_2) = \frac{0.6}{\sqrt{(0.6^2 + 0.8^2)} \times \sqrt{(0.6^2 + 0.3^2 + 0.3^2)}} = \frac{0.6}{\sqrt{0.54}} \approx 0.816$

$\cos\gamma = \text{Sim}(Q, D_3) = \frac{0.44}{\sqrt{(0.6^2 + 0.8^2)} \times \sqrt{(0.6^2 + 0.1^2 + 0.3^2)}} = \frac{0.44}{\sqrt{0.46}} \approx 0.649$

余弦相似度越接近 1，相关度越高。且 $0.943 > 0.816 > 0.649$ ，文档 D_1 与查询向量 Q 的相关度更高。

6.1.4 分析二者的优劣

	内积相似度	余弦相似度
优点	计算简单	对内积相似度进行了归一化处理，避免了文档长度对相似度量度的影响。
缺点	文档越长，文档对应的向量权重就会越大。内积相似度会在较长文档上得到较大相似度。	在一些需要考虑向量长度信息的应用场景中表现不佳。

6.2 万维网链接结构图

6.3 HITS算法

6.4 PageRank算法

6.5 搜索引擎评价指标