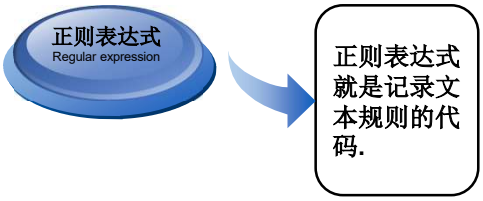
正则表达式入门

正则表达式

正则表达式——概念

百度百科：

在计算机科学中，是指一个用来描述或者匹配一系列符合某个句法规则的字符串的单个字符串。



正则表达式——作用

1. 测试字符串的某个模式。例如，可以对一个输入字符串进行测试，看该字符串是否是一个电话号码或一个信用卡号码。*数据有效性验证。*
2. 替换文本。可以在文档中使用一个正则表达式来标识特定文字，然后可以全部将其删除，或者替换为别的文字。
3. 根据模式匹配从字符串中提取一个子字符串。可以用来在文本或输入字段中查找特定文字。

正则表达式

- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

正则表达式——语法

- 组成：
  - 正则表达式由一些普通字符和一些**元字符**（meta-characters）组成。
  - 普通字符包括大小写的字母和数字，在正则表达式中具有一定的特殊含义，这些字符叫做**元字符**。
  - eg:0\d\d-\d\d\d\d\d\d\d\d
  - \d：匹配一个数字字符。等价于 [0-9]。

正则表达式——常用元字符

.	• 匹配除换行符以外的任意字符
\w	• 匹配字母或数字或下划线
\s	• 匹配任意的空白符
\d	• 匹配数字
\b	• 匹配单词的开始或结束
^	• 匹配字符串的开始
$	• 匹配字符串的结束
\n	• 匹配一个换行符

Regular Expressions: Anchors ^ $

Pattern	Matches
\.$	The end.
.$	The end? The end!

正则表达式——常用反义字符

\W	• 匹配任意不是字母，数字，下划线，汉字的字符
\S	• 匹配任意不是空白符的字符
\D	• 匹配任意非数字的字符
[^x]	• 匹配除了x以外的任意字符
[^aeiou]	• 匹配除了aeiou这几个字母以外的任意字符

正则表达式——常用限定符

{n}	• 重复n次
{n,}	• 重复n次或更多次
{n,m}	• 重复n到m次
*	• 重复零次或更多次
+	• 重复一次或更多次
?	• 重复零次或一次

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n

正则表达式——“？”用法

*?	• 重复任意次，但尽可能少重复
+?	• 重复1次或更多次，但尽可能少重复
??	• 重复0次或1次，但尽可能少重复
{n,m}?	• 重复n到m次，但尽可能少重复
{n,}?	• 重复n次以上，但尽可能少重复

正则表达式-".*"和".*?"

- .* 匹配任意字符0或者多次(大于等于0次)
- .*? 满足条件的情况只匹配一次，即最小匹配.
- 比如：比如：<H1>Chapter 1 - 介绍正则表达式 </H1>
  - 使用/<.*>/匹配的结果为：H1>Chapter 1 - 介绍正则表达式 </H1。
  - 使用/<.*?>/匹配结果为：H1。

正则表达式——[](1)

- [ ] 、[m-n]：
  - 匹配括号中的任何一个字符。例如正则表达式r[aou]t匹配rat、rot和rut，但是不匹配ret。
  - 可以在括号中使用连字符-来指定字符的区间，例如正则表达式[0-9]可以匹配任何数字字符；
  - 还可以制定多个区间，例如正则表达式[A-Za-z]可以匹配任何大小写字母

正则表达式——[](2)

- [^m-n]：
  - 要想匹配除了指定区间之外的字符——也就是所谓的补集——在左边的括号和第一个字符之间使用^字符，例如正则表达式[^269A-Z] 将匹配除了2、6、9和所有大写字母之外的任何字符。
  - [^st]he，所有含he的字符串，除了she和the以外。

Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

- Negations [^Ss]

Pattern	Matches	
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[e^]	either e or ^	Look here
a^b	The pattern a carat b	Look up a^b now

Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

Photo D. Fletcher

正则表达式——优先级顺序

1.\	• 转义符
2.(),(?:),(?=),[]	• 圆括号和方括号
3.*, +, ?, {n}, {n,}, {n,m}	• 限定符
4.^, $, \anymetacharacter	• 元字符
5.|	• “或”操作

Regular Expressions: Negation in Disjunction

•Negations [^Ss]

Pattern	Matches	
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither ‘S’ nor ‘s’	I have no exquisite reason”
[e^]	either e or ^	Look here
a^b	The pattern a carat b	Look up a^b now

圆括号（）

•圆括号()是组，主要应用在限制多选结构的范围/分组/捕获文本/环视/特殊模式处理

- (ab){1,3},就表示ab一起连续出现最少1次，最多三次。如果没有括号的话，ab{1,3}?
- (abc|bcd|cde)，表示这一段是abc、bcd、cde三者之一，顺序也必须一致
- (abc)？ 表示这一组要么一起出现，要么不出现，出现那则按顺序出现

圆括号（）

•圆括号()是组，主要应用在限制多选结构的范围/分组/捕获文本/环视/特殊模式处理

- （？：abc）表示找到一样abc的一组，但是不记录，不保存到变量中，否则可以通过变量中，否则可以通过x取第几个括号所匹配项

例：(aaa)(bbb)(ccc)(?:ddd)(eee)可以用1获取(aaa)匹配到的内容，而1获取(aaa)匹配到的内容，而3则获取到了(ccc)匹配到的内容，而4则获取的是由(eee)匹配到的内容，因为前一对括号没有保存变量

•a(?=bbb)顺序环视 表示a后面必须紧跟3个连续的b

•(?i)，(?i:X)不区分大小写 (?s:.*)跨行匹配，可以匹配回车符

•能够完全匹配字符串“c:\rapidminer\lib\plugs" 的正则表达式包括:( )

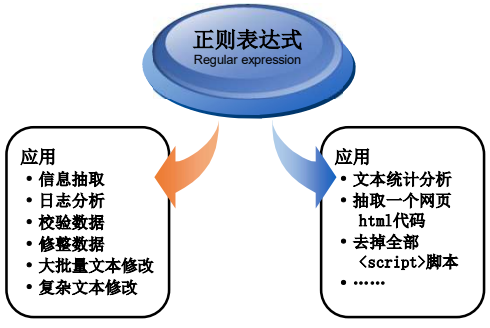
•A.“c:\rapidminer\lib\plugs”

•B.“ c:\\rapidminer\\lib\\lplugs”

•C.“(?i)C:\\RapidMiner\\Lib\\Plugs”

•D.“(?s)C:\\RapidMiner\\Lib\\Plugs”

正则表达式——应用



经典正则表达式实例

^[A-Za-z]+$	由26个字母组成的字符串
^[A-Za-z0-9]+$	由26个字母和数字组成的字符串
^-?\d+$	整数形式的字符串
^[0-9]*[1-9][0-9]*$	正整数形式的字符串
[1-9]\d{5}	中国境内邮政编码，6位
[\u4e00-\u9fa5]	匹配中文字符
\d{3}-\d{8}|\d{4}-\d{7}	国内电话号码，010-68913536

正则表达式——应用

E-mail:

^([a-zA-Z0-9]+[_|\-|\.]?)*[a-zA-Z0-9]+@([a-zA-Z0-9]+[_|\-|\.]?)*[a-zA-Z0-9]+(\.[a-zA-Z]{2,3})+$

- 验证用户密码，长度在6~18 之间，只能包含英文和数字?

```
^[A-Za-z0-9]{6,18}$
```

正则表达式应用-替换

- 正则表达式:

  s/([0-9]+)/<\1>

第一步：(…)表示模式

第二步："\1"可以用圆括号中的项相匹配的任何符号串来替换。

the 35 boxes替换为the <35> boxes

正则表达式应用-替换

- 正则表达式:
  - /the (.*)er they (.*) , the \ 1er they \2/

寄存器
(register)

The bigger they were, the bigger they were

用户$_1$: Men are all alike.
ELIZA$_1$: IN WHAT WAY.
用户$_2$: They're always bugging us about something or other.
ELIZA$_2$: CAN YOU THINK OF A SPECIFIC EXAMPLE.
用户$_3$: Well, my boyfriend made me come here.
ELIZA$_3$: YOUR BOYFRIEND MADE YOU COME HERE.
用户$_4$: He says I'm depressed much of the time.
ELIZA$_4$: I AM SORRY TO HEAR YOU ARE DEPRESSED.

ELIZA进行一连串的正则表达式的替换，每个正则表达式与输入行中的某个部分相匹配，然后替换并改变它们。首先的替换是把所有的*my*替换为*YOUR*，把所有的*I'm*替换为*YOU ARE*，等等。然后的替换是查找输入中相关的模式，并产生出适合的输出。下面是一些例子：

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

正则表达式应用-替换

- 题目：
  - 已知正则表达式的模式为："/the (.*)er they were， the \1er they will be/"，下面哪项不满足该模式？

a. the bigger they were, the bigger they will be
b. the bigger they were, the faster they were
c. the Xer they were, the Xer they will be

Python的正则表达式

- re库是Python的标准库，主要用于字符串匹配。(不需要安装)
  - 调用方式：import re
  - 正则表达式的表示类型
    raw string类型（原生字符串类型）：不包含转义符的字符串，原生字符串中的\不会被解释成转义符
  - re库采用raw string类型表示正则表达式，表示为：r'text'
    例如：r'[1-9]\d{5}'
    r'\d{3}-\d{8}|\d{4}-\d{7}'

```
1.str1 = r'I\'m a great coder!'
2.print(str1)
输出结果：I\'m a great coder!
```

```
1.str1 = 'I\'m a great coder!'
2.print(str1)
输出结果： I'm a great coder!
```

Python的正则表达式库re

```
regex = re.compile(pattern, flags=0)
```

将正则表达式的字符串形式编译成正则表达式对象

- pattern ： 正则表达式的字符串或原生字符串表示
- flags ： 正则表达式使用时的控制标记

```
>>> regex = re.compile(r'[1-9]\d{5}')
```

Python的正则表达式库re

Re库主要功能函数

函数	说明
re.search()	在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象
re.match()	从一个字符串的开始位置起匹配正则表达式，返回match对象
re.findall()	搜索字符串，以列表类型返回全部能匹配的子串
re.split()	将一个字符串按照正则表达式匹配结果进行分割，返回列表类型
re.finditer()	搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象
re.sub()	在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

- Match对象的属性

属性	说明
.string	待匹配的文本
.re	匹配时使用的patter对象（正则表达式）
.pos	正则表达式搜索文本的开始位置
.endpos	正则表达式搜索文本的结束位置

- Match对象

方法	说明
.group(0)	获得匹配后的字符串
.start()	匹配字符串在原始字符串的开始位置
.end()	匹配字符串在原始字符串的结束位置
.span()	返回(.start(), .end())

Python的正则表达式库re

```
re.search(pattern, string, flags=0)
```

- flags ： 正则表达式使用时的控制标记

常用标记		说明
re.I	re.IGNORECASE	忽略正则表达式的大小写，[A-Z]能够匹配小写字符
re.M	re.MULTILINE	正则表达式中的^操作符能够将给定字符串的每行当作匹配开始
re.S	re.DOTALL	正则表达式中的.操作符能够匹配所有字符，默认匹配除换行外的所有字符

基于python的正则表达式

- 在Python中使用正则表达式，需要使用import re 导入re函数库。还需要一一个用于搜索的词汇链表;我们再次使用词汇语料库c
```
>>> import re
>>> wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
```

- 让我们使用正则表达式《ed$》查找以ed结尾的词汇。函数re.search(p, s)检查字符串s中是否有模式p。我们需要指定感兴趣的字符的符号，
```
>>> [w for w in wordlist if re.search('ed$', w)]
['abaissed', 'abandoned', 'abased', 'abashed', 'abatised', 'abed', 'aborted', ...]
```


- 通配符“.”匹配任何单个字符。假设我们有一个8个字母组成的词的字谜室, j是其第三个字母，t是其第六个字母。空白单元格中的每个地方,我们用一个句点·
```
>>> [w for w in wordlist if re.search('^.j..t..$', w)]
['abjectly', 'adjuster', 'dejected', 'dejectly', 'injector', 'majestic', ...]
```



```
>>> rex = r"a.d"   # 正则表达式文本
>>> original_str = "and"  # 原始文本
>>> pattern = re.compile(rex)  # 正则表达式对象
>>> m = pattern.match(original_str)  # 匹配对象
>>> m
<_sre.SRE_Match object at 0x101c85b28>

# 等价于
>>> re.match(r"a.d", "and")
<_sre.SRE_Match object at 0x10a15dcc8>
```



```
>>> re.match(r"a.c", "abc").group()
'abc'
>>> re.match(r"a.c", "abcef").group()
'abc'
>>> re.match(r"1\.2", "1.2").group()
'1.2'
>>> re.match(r"a[0-9]b", "a2b").group()
'a2b'
>>> re.match(r"a[0-9]b", "a5b11").group()
'a5b'
>>> re.match(r"a[.*?]b", "a.b").group()
'a.b'
>>> re.match(r"abc[^\w]", "abc!123").group()
'abc!
```


- *group* 方法返回原字符串(abc)中与正则表达式相匹配的那部分子字符串(abc)，提前是要匹配成功 *match* 方法才会返回 *Match* 对象，进而才有*group*方法。

```
m = re.match(r"(\d+)(\w+)", "123abc")
# 分组 0，匹配整个正则表达式
>>> m.group()
'123abc'
#等价
>>> m.group(0)
'123abc'
# 分组1，匹配第一对括号
>>> m.group(1)
'123'
# 分组2，匹配第二对括号
>>> m.group(2)
'abc'
```


推荐两个在线测试正则表达式的工具:
RegexBuddy
Javascript正则表达式在线测试工具

网页爬虫的例子

- 人民网 http://jhsjk.people.cn/result/1?form=706&else=501
- 网页分析（实际分析

Thank You !