

通配符是指按照一定的模式匹配一个或多个真正的字符，比如Notep?d可以对应Notepad, Noteped，是一种模 糊检索的方式。在Linux和类Unix系统中的Shell命令行中，通配符被广泛使用。与简单的通配符不同，正则表达 式要强大的多。正则表达式于1960s由Thompson引入Unix系统的标准文本编辑器，自此以后正则表达式被广泛 地应用于各种Unix或类Unix系统的工具中。正则表达式遵循一定的规范，最著名的是从Perl衍生出来的 PCRE（Per Compatible Regular Expression）规范和POSIX（Portable Operating System Interface for Unix）规 范定义的正则规范。目前，各个编程语言都或多或少地支持一定的程度的正则表达式，但是也有一些自己特有 的不同之处。

正则表达式是处理文本和字符串的利器。假设你想检查一个邮箱地址是否合法，如果不用正则表达式，请问你 会用什么来处理呢？自己编写一段复杂的字符串匹配的代码吗？利用正则表达式，我们可以使用一句不太长的 字符，完成一个很复杂的功能。但是，过长的正则表达式晦涩难懂，即使写出这些正则表达式，过段时间后也 不见得能够解释清楚。

带着问题学习，能够有效地增加高级知识，更有针对性，学习效果会更好。为了完成下面的讲述，我们先来看 几个问题。

我们使用的文本是医疗领域的50000句中英双语句对，中间以空格分隔，有以下的整理任务。

将中英文句对拆分

如果中英文中间没有空格。

将中英文句对拆分，拆分规则是英文中没有中文

将中英文句对拆分，拆分规则是中文中没有英文

拆分以后的中英文句对以" ||| "分隔。想象一下我们有以下的检索任务：

数字，包括整数和小数 英文缩写，骆驼拼写等特殊单词 带有连字符的英文单词，包括化合物名字

如果搞懂了以上的集个案例，那我们也就入门了通配符和正则表达式。



Word中的通配符在查找替换时很有用。

1. ?, \*

?代表一个单独的字符，例如s?t可以找到sat, sit, set等开头为s结尾为t的长度为3的字符串

\*代表任意个数的字符，例如s\*t可以找到st, sit, secret, serpent, sailing boat等

2. @

@代表重复前一个字符一次或者多次，例如lo@t可以找到lot, loot, 非贪婪

3. <, >

<代表一个单词的开始，而>代表一个单词的结束。他们本身不匹配任何单词，但是能够指明匹配的位 置。例如<s\*t>可以匹配任何以s开头以t结尾的片段：tea‐set中的set，但是不能匹配toolset中的 set，因为不是单词边界。

<, >可以单独使用，例如ful@>可以匹配full, wilful中的ful，但是不能匹配wilfully中的full。

4. []

用[]表示一个集合中的任意一个元素，[abc]会匹配a, b, c中的任意一个字符。[]也能够支持区间查找， 例如[A‐Z]可以匹配所有的英文大写字母，[A‐Za‐z]可以匹配所有的英文字母。

5. \

如果你想查找任意有特殊含义的字符，例如通配符中的?, \*, @, 应当在前面加上\进行转义，例如\?, \\*,

\@。如果想要查找\本身，应当使用\\。

列举通配符查找中的特殊含义字符： [ ] { } < > ( ) ‐ @ ? ! \* \

6. [!]

用[!]表示任意一个非集合元素的字符，[!abc]会匹配除了a, b, c以外的一个任意字符。[!]也能够支持 区间查找，例如[!A‐Z]可以匹配所有非英文大写字母的字符。

7. {}

用{n}表示重复前面的字符n次，{n,}表示重复前面的字符不少于n次，{n,m}表示重复前面的字符不少于

n次，不多于m次。例如：[deno]{4}可以匹配done, node, eden，a{2,3}可以匹配aa, aaa

8. ()

用()表示捕获组，例如(John) (Smith)，其第一个捕获组\1指的是John，第二个组\2指的是Smith。 这在替换中非常有用，例如，将JavaScript, PowerShell替换为Java Script, Power Shell, 可以这样 写：

查找：<([A‐Z][a‐z]@)([A‐Z][a‐z]@)> 替换：\1 \2

这在解析html标签时也十分有用，例如给<title>正则表达式</title>脱括号： 查找：\<([!<>]@)\>([!<>^13]@)\</\1\>

替换：\2

9. ^

使用^代表acsii字符转义和一些特殊的字符，例如制表符^t, 可以在Word的特殊格式中找到。有一些例如

^p段落标记不能与通配符同时使用，因此需要用unicode编码。

1. 贪婪，非贪婪

贪婪的含义是：尽可能地匹配全部的字符串。Word的查找是从光标位置向后一位一位地查找，使用\*, @ 时，一旦匹配表达式，则报告，因此，\*, @是非贪婪的。与之不同的是，{n,}则会侵吞遇到的全部字符 串，并不再回溯。

要匹配looootloooot, 由于\*, @非贪婪，故应当对匹配内容进行精确描述： l\*t或者l[lot]@t只能够匹配前面的loooot, 而l\*t>或者l[lot]@t>则可以匹配全部。 l[lot]{1,}t不能匹配，因为l[lot]{1,}就已经侵吞了全部的字符串，没有机会留给最后一位的t。

2. 回溯

使用a@ab和字符串aaaaab匹配，首先侵吞全部的a，然后吐出一位，最后匹配成功

3. 字符编码

使用^后面跟上数字，可以表示一位ASCII字符。例如^13代表的就是回车符\r, 也就是Word里面的段落标 记^p。ASCII有效范围[^1‐^128]。

此外还支持unicode范围查找，比如匹配中文：[一‐龥]，即unicode编码[\u4e00‐\u9fa5]。

Unicode字符：

Unicode字符列表: https://zh.wikipedia.org/wiki/Unicode%E5%AD%97%E7%AC%A6%E5%88



%97%E8%A1%A8

Unicode/Character reference: https://en.wikibooks.org/wiki/Unicode/Character\_reference



/0000-0FFF

Unicode 10.0 Character Code Charts: [https://www.unicode.org/cha](http://www.unicode.org/charts)rts



世界文字大全，Unicode 字符集: [http://www.qqxiuzi.cn/zh/unicode-zifu.php?plane=0&ks=0000&](http://www.qqxiuzi.cn/zh/unicode-zifu.php?plane=0&amp;ks=0000) js=0FFF



Word中的查找与替换通配符详解: https://wenku.baidu.com

/view/3c2f6b59ad02de80d4d8406b.html?re=view

Finding and replacing characters using wildcards: https://wordmvp.com/FAQs/General

/UsingWildcards.htm

自己理解并能够书写出正确的符合自己业务逻辑的表达式，不应当盲目迷信网上的解决方案

1. 中英文以空格分隔

([!^13 ]@) ([!^13]@^13)

2. 英文中没有中文

([!^13]@[一‐龥])([!一‐龥]@^13)

3. 中文没有英文

([!^13 A‐Za‐z]@)([A‐Za‐z][!^13]@^13)

4. 数字，包括整数和小数

<[0‐9]@> <[0‐9]@.[0‐9]@>

由于Word中没有匹配0个或者1个的概念，因此只能分两步来写

5. 英文缩写，骆驼拼写等特殊单词

<[A‐Z]{2,}> <[A‐Z][a‐z]@[A‐Z][a‐z]@>

由于Word中没有重复表达式多次的概念，因此只能匹配固定个数的单词拼接的骆驼拼写

6. 带有连字符的英文单词，包括化合物名字

<[A‐Za‐z]@\‐[A‐Za‐z]@>

由于Word中没有重复表达式多次的概念，因此只能匹配固定个数连字符；由于没有匹配0个或1个的概 念，因此如果引入数字，需要指定数字与英文的相对位置



相比于通配符，正则表达式更为强大。

1. 元字符

# 代

**说明**

**码**

. 匹配除换行符\n以外的任意字符

# 代

**说明**

**码**

匹配字母或数字或下划线，例如\w{6}可以匹配6个字符的单词。python3.6中可以\w可以匹

\w 配字母(unicode支持)或数字或下划线或汉字

\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，类似通配符中的<,>标记

^ 匹配字符串的开始，在多行匹配中标记了一行的开始

$ 匹配字符串的结束，在多行匹配中标记了一行的结束 例如：句子有大写首字母，可以使用^[A‐Z]

2. 重复

# 代码 说明

\* 重复零次或更多次

+ 重复一次或更多次

? 重复零次或一次

{n} 重复n次

{n,} 重复n次或更多次

{n,m} 重复n到m次

3. 字符类

使用[]表示任意一个集合内的元素，类似Word通配符，例如[abcde]表示a, b, c, d, e中的任意一个， [a‐z]表示任意一个小写英文字母

4. 反义

使用[^]表示任意一个非集合内的元素，类似Word通配符[!]，例如[^abcde]表示除了a, b, c, d, e的任 意一个字符。此外，还有一些元字符的反义：

# 代码 说明

\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

\B 匹配不是单词开头或结束的位置

使用元字符及元字符反义时，一定要首先知道其能够表示的范围。不同的语言所支持正则的元字符可能

有不同的表示范围，这一点在1中介绍了

5. 分组

使用()来指定子表达式。与Word通配符中的()不同，子表达式在重复时重复的是表达式而不是捕获的 组。具体的可以看这样的例子：对lotlet

Word : <(l[eo]t)@>不能匹配，<l[eo]tl[eo]t>可以匹配 Regex: \b(l[eo]t)+\b可以匹配 正则和Word一样可以引用()捕获的组

6. 分枝条件

使用|来表示分支条件，也即或的概念。例如，要匹配1900-2099全部的年份，如果使用Word通配符要分 两种情况书写，但是使用正则就方便得多：

(19|20)[0‐9]{2}

7. 字符转义

这个概念在Word通配符中也存在。需要在想要查找的元字符之前加上\ 列举正则中需要转义的字符： [] { } ( ) - \ |



1. 零宽断言

有一种特殊的分组并不会被捕获，这就是零宽断言。零宽断言在查找替换时很有用。下面先给出最常用 的几种分组语法：

(?=exp) 匹配exp前面的位置

|  |  |  |
| --- | --- | --- |
| **分类** | **代码** | **说明** |
| 捕获 | (exp) | 匹配exp,并捕获文本到自动命名的组里 |
| - | (?<name>exp) | 匹配exp,并捕获文本到名称为name的组里，也可以写成(?'name'exp) |
| - | (?:exp) | 匹配exp,不捕获匹配的文本，也不给此分组分配组号 |
| 零宽断 | | |
| 言 |  |  |
| - | (?<=exp) | 匹配exp后面的位置 |
| - | (?!exp) | 匹配后面跟的不是exp的位置 |
| - | (?<!exp) | 匹配前面不是exp的位置 |

# 分类 代码 说明

注释 (?#comment)

这种类型的分组不对正则表达式的处理产生任何影响，用于提供注释让 人阅读

首先来看捕获。我们已经熟悉了最简单的捕获(exp)，这里还有两个：(?<name>exp)和(?:exp)。例 如：(?<name>[abc]) \k<name>可以匹配a a, b b, c c；而(?:a|b)c虽然和(a|b)c功能一样，但是不 会捕获组，这使得在运行正则时开销更小

再来看零宽断言。q(?=u)可以匹配所有后面紧跟着u的q，而(?<=q)u可以匹配所有前面是q的u。具体一 点，我们想要找到所有以.结尾的英文单词，并在.前加上空格，我们可以这样写：

查找：\b([A‐Za‐z]+)(?=\.) 替换：$1

可以看到我们并没有捕获\.。当想要实现的功能变得复杂时，使用零宽断言可以简化书写。 而负向零宽断言(?!exp) (?<!exp)则与零宽断言相反，**不会**匹配后面/前面为exp的内容，例如： (?<=[一‐龥])[A‐Za‐z]+(?=[一‐龥]) 匹配前后都是汉字的英文单词 (?<![一‐龥A‐Za‐z])[A‐Za‐z]+(?![一‐龥A‐Za‐z]) 匹配前后都不是汉字的英文单词

2. 贪婪与懒惰 与Word中不同，正则表达式支持贪婪或者懒惰匹配

# 代码 说明

\*? 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

例如，要匹配looootloooot，l.\*t会匹配looootloooot，而l.\*?t则只会匹配loooot。

例如，要匹配重复片段，\b(.+?)\b( \*\b\1\b)+，由于无法确定片段的组成，因此只能借助非贪婪和 单词边界判断。

3. 字符编码

使用\u后面跟上4位的16进制数，可以匹配一个unicode字符，比如匹配中文：[一‐龥]，即unicode编码

[\u4e00‐\u9fa5]。字符的unicode编码可以在上一章节的链接中查找到。

4. 平衡组

平衡组可以对配对括号进行检验，一个很实在的应用就是返回一个字符串里最长的html标签。

正则表达式30分钟入门教程: <http://deerchao.net/tutorials/regex/regex.htm>

正则表达式菜鸟教程: <http://www.runoob.com/regexp/regexp-syntax.html>

Notepad++: https://notepad-plus-plus.org/

熟练掌握正则表达式的方法还是多写多练，多读

1. 中英文以空格分隔

^([^ ]+)

2. 英文中没有中文

(?<=[一‐龥])(?=[^一‐龥]\*$)

3. 中文没有英文

^([^A‐Za‐z ]+)

1. 数字，包括整数和小数

[0‐9]+(\.[0‐9]+)?

以上的正则会匹配英文代号如H7N9中的7和9，怎样解决？(\b要考虑中文)

2. 英文缩写，骆驼拼写等特殊单词

\b[A‐Z]{2,}\b \b([A‐Z][a‐z]+){2,}\b

3. 带有连字符的英文单词，包括化合物名字

\b([A‐Za‐z0‐9]+|[0‐9]+(,[0‐9]+)\*)(\‐[A‐Za‐z0‐9]+|\‐[0‐9]+(,[0‐9]+)\*)+\b

可以匹配： 5‐year, long‐term, Laurence‐Moon‐Biedl

4. 叠词

\b([A‐Za‐z]+)\1([A‐Za‐z]+)\2\b \b(?<w1>[A‐Za‐z]+)\k<w1>(?<w2>[A‐Za‐z]+)\k<w2>\b