

Lenguajes formales y expresiones regulares

Dr. José Lázaró Martínez Rodríguez

Expresiones regulares

<http://xkcd.com/208/>



Regex

- En cómputo teórico y teoría de lenguajes formales, una **expresión regular**, o expresión racional (regex/regexp), es una secuencia de caracteres que conforma un patrón de búsqueda.
- Se utilizan principalmente para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones.

Regex

- Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto.
- Las expresiones regulares proporcionan una manera muy flexible de buscar o reconocer cadenas de texto.
 - Por ejemplo, el grupo formado por las cadenas Handel, Händel y Haendel se describe con el patrón "H(a|ä|ae)ndel".

Entendiendo Regex

- Muy potentes y bastante crípticos
- Divertidas una vez que las entiendes
- Las expresiones regulares son un lenguaje en sí mismas
- Un lenguaje de "caracteres marcadores" - programación con caracteres
- Es una especie de lenguaje de la "vieja escuela" - compacto

Expresiones regulares

- Una expresión regular es un patrón que coincide con algún texto regular (predecible).
- Las expresiones regulares se utilizan en muchas utilidades de Unix.
 - Como apps: grep, sed, vi, emacs, awk, ...
- La forma de una expresión regular:
 - Puede ser texto plano ...
 - > grep unix file (coincide con todas las apariciones de unix)
- También puede ser un texto especial ...
 - > grep '[uU]nix' file (coincide con unix y Unix)

Construcción de expresiones regulares

- Específicamente, las expresiones regulares se construyen utilizando los operadores unión, concatenación y cerradura de Kleene.
 - Toda expresión regular tiene algún autómata finito asociado. [Ya veremos]
- Escapar los caracteres reservados es crucial
 - `/(i.e. /` no es válido porque `(` debe estar cerrado
 - Sin embargo, `/(i\.e\N. /` es válido para encontrar `'(i.e. '`
- Los caracteres reservados incluyen:
- `. * ? + () [] { } / \ |`
 - Además, algunos caracteres tienen un significado especial en función de su posición en el enunciado

Coincidencia de texto (Matching)

- Una RegEx puede coincidir con texto plano
 - `/Dan/` `/*Perl*/`
 - Pero esto coincidirá con `Dan`, `Danny`, `Daniel`, etc...
- Comparación de texto completo con anclas (anchors)
 - Puede querer coincidir con una línea completa (o cadena)
 - ex. `^Dan$`
- Esto sólo coincidirá con `Dan`
- `^` se ancla al principio de la línea
- `$` se ancla al final de la línea

Coincidencia de texto (Matching)

- Orden de los resultados
 - La búsqueda comenzará al principio de la cadena
 - Esto puede ser alterado, no preguntes todavía
- Cada carácter es importante
 - Cualquier texto plano en la expresión se trata literalmente
 - No se descuida nada (el cierre no cuenta)
 - `/s/` no es lo mismo que `/ s/` (`/regex/` usado en Perl)
- Mucho más fácil de escribir que de depurar

Cadenas de clases

- Permite especificar sólo algunos caracteres permitidos
- [dofZ] sólo coincide con las letras d, o, f, Z
 - Si tienes una cadena 'dog' entonces /[dofZ]/ solo empata 'd' incluso aunque 'o' también está en la clase
- Así que esta expresión se puede enunciar como "coincide con una de las d, o, f, o Z".

Cadenas de clases

<https://regex101.com/>

regular expressions 101

[@regex101](#) [\\$ donate](#) [❤ sponsor](#) [✉ contact](#) [🚩 bug reports & feedback](#) [? wiki](#) [📖 whats new?](#)

</>

SAVE & SHARE

📄 Save Regex

🔗 % + S

FLAVOR

</> PCRE2 (PHP >=7.3)

</> PCRE (PHP <7.3)

</> ECMAScript (JavaScript)

</> Python 2.7

</> Golang (RE2)

FUNCTION

>_ Match

✂ Substitution

🧪 Unit Tests

TOOLS

📄 Code Generator

REGULAR EXPRESSION

2 matches (~167ms)

`

[dofZ]

`

gm

📄

TEST STRING

dog

SUBSTITUTION

success (~0ms)

insert your replacement value here

g

EXPLANATION

▼ ` [dofZ] ` gm

▼ Match a single character present in the list below

[dofZ]

dofZ matches a single character in the list dofZ (case sensitive)

- Global pattern flags

MATCH INFORMATION

Match 1

Full match 0-1 d

Match 2

Full match 1-2

QUICK REFERENCE

Search reference

A single char... [abc]

Cadenas de clases

- `[A-Za-z]` coincide cualquier letra
- `[a-fA-F0-9]` coincide cualquier carácter hexadecimal
- `[^*$/\\]` coincide con todo menos `*`, `$`, `/`, o `\`
 - El `^` al frente del carácter de clase especifica 'not'
 - En un carácter de clase, solo necesitas escapar `\ (] - ^`
 - Y puede ser que punto `(.)` ya que es un comodín

Ejemplos

Erick nació en 1997 en Reynosa

David nació en 1998 en Victoria

Lucía nació en 2001 en Tampico

¿Algo en común?

¿Qué patrón describe el año?



`/[0-9][0-9][0-9][0-9]/`

O qué tal?

`/1[0-9][0-9][0-9]/`

¿Algún problema?

Ya no abarcaría los nacidos
después de 1999

Ejemplos

Erick nació en 1997 en Reynosa

David nació en 1998 en Victoria

Lucía nació en 2001 en Tampico



/[12][09][0-9][0-9]/

¿Algo en común?

¿Otra forma de representar?

¿Algún problema?

Cubre lo necesario para este problema 😊
pero luego veremos cómo abarcar la ciudad

Ejemplos

El vino cuesta 100.50

Mi hamburguesa costó 123.43

120.50 por boleto de autobús

¿Algo en común?



¿Algún patrón que describa
solo estos precios?

`/[0-9][0-9][0-9].[0-9][0-9]/`

¿Algún problema?

No se está escapando el símbolo '.'

`/[0-9][0-9][0-9]\.[0-9][0-9]/`

Ejemplos

Identifique palabras de tres letras

Oso
Ave
casa
río

¿Algún patrón que describa
esta búsqueda?

/[a-z][a-z][a-z] /

¿Algún problema?

No abarca mayúsculas ni acentos
No debe contener espacio

/[A-Za-zá-ú][A-Za-zá-ú][A-Za-zá-ú]/

¿Algún problema?

Ejemplos

¿Algún patrón que describa
esta búsqueda?

Identifique palabras de tres letras

`/^[A-Za-zá-ú][A-Za-zá-ú][A-Za-zá-ú]$/`

Oso
Ave
casa
río

¿Algún problema?

No por ahora 😊

Cadenas de clase

Caracteres de clase coinciden con caracteres específicos

- `\d` coincide con un solo dígito
- `\w` coincide con caracteres de palabra (A-Z, a-z, `_`)
- `\b` coincide con el límite de una palabra `/\bword\b/`
- `\s` coincide con caracter espacio en blanco (spc, tab, newln)
- `.` Comodín/wildcard coincide con todo excepto nueva línea
 - Usar con cuidado, puede obtener lo que sea!
- Para que coincida con "cualquier cosa menos...", ponga en mayúsculas la clase
 - i.e. `\D` coincide con cualquier cosa que no sea un dígito

Ejemplos

Erick nació en 1997 en Reynosa

David nació en 1998 en Victoria

Lucía nació en 2001 en Tampico

¿Otra forma de representar?



`/\d\d\d\d/`

¿Algo en común?

Ejemplos

Identifique palabras de tres letras

Oso

Ave

casa

río

¿Algún patrón que describa
esta búsqueda?

/\w\w\w/

¿Algún problema?

Complicado detectar acentos

àèìòùÀÈÌÒÙáéíóúýÁÉÍÓÚÝâêîôûÂÊÎÔÛãñõÃÑÕäëïöüÿÄËÏÖÜŸçÇßÐøÅåÆæœ

/^...\$/

¿Algún problema?

Va a detectar cualquier conjunto de tres símbolos

Ejemplos

Si tenemos /Patin[ae]s/

El patrón busca: Patin,

Después una de las letras [ae],

Después s

Lo que coincide con cualquiera
de las palabras

Patinas
Patines

Ejemplos

- Coincidir

ear, eye, etc `/e\w\w/`

‘1, 2, 3 strikes!’ `/\s\d/` Empata a? ‘2’

‘1, 2, 3 strikes!’ `/[\s\d]/` Empata a? ‘1’

889-112-18-34 Sería? `/[\d\d\d-\d\d\d-\d\d-\d\d]/`

Hay una mejor manera
Se debe escapar el ‘-’

Repeticiones

- Indica la cantidad de veces que se puede repetir un caracter
 - `CaracterARepetir{#veces}`
 - o también `CaracterARepetir{#minVeces, #maxVeces}`
- Rango de ocurrencias
- `/\d{2,3}/`
 - Coincide cualquier número de 10 a 999
- `/\w{5,}/`
 - Coincide cualquier nombre mas largo de 5 letras
- `/\d{9}/`
 - Coincide exactamente 9 dígitos

Ejemplos

Si tenemos `/\w{4}\d{6}/`

El patrón busca cuatro letras: `\w{4}`,

Después 6 dígitos `\d{6}`,

MARJ970712
RIAA980322

La primer parte del CURP

Si tenemos `/[Dd]istancia: \d{2,4}km/`

El patrón busca una letra: `[Dd]`,

El patrón busca la cadena: `istencia:`,
(note el espacio)

Después de dos a cuatro dígitos `\d{2,4}`,

Al final la cadena: `km`

distancia: 20km
Distancia: 1000km
Distancia 1km

Ejemplos

¿Otra forma de representar?

Erick nació en 1997 en Reynosa
David nació en 1998 en Victoria
Lucía nació en 2001 en Tampico

`/\d{4}/`

El vino cuesta 100.50
Mi hamburguesa costó 123.43
120.50 por boleto de autobús

`/\d{3}\.\d{2}/`

Oso
Ave
casa
río

`/^[A-Za-zá-ú]{3}$/`

Repeticiones

- Cuantificadores generales $*+?$
- Otros caracteres especiales
- $/\textcolor{red}{d}^*/$
 - Coincide cualquier número (de cero hasta n dígitos)
- $/\textcolor{red}{w}^+ /$
 - Coincide uno o mas caracteres
- $/\textcolor{red}{w}? /$
 - Coincide uno o ningún carácter (zero)

Repeticiones

- Coincide con el patrón
- `/\d*/`
 - Coincide con: 1, 2, 3, 12, 1231231231231123123123
- `/\d+/`
 - Coincide con: 1, 2, 3, 12, 1231231231231123123123

¿Alguna diferencia?

No muy evidente a menos que
se encuentre concatenado con
más elementos

Ejemplos

Si tenemos `/file\d*/`

El patrón busca cuatro letras: **file**

Después cero o mas dígitos **\d***,

```
file  
file100000000  
file2
```

Si tenemos `/file\d+/`

El patrón busca cuatro letras: **file**

Después uno o mas dígitos: **\d+**,

```
file  
file100000000  
file2
```

Y el patrón `/file\d?/` qué busca?

Ejercicios

- Una expresión para
- Análisis de etiquetas de enlaces HTML
 - `<`, opcionalmente seguido de espacio en blanco, seguido de `a`, seguido por espacio en blanco, seguido por `href`, opcionalmente seguido por espacio en blanco, seguido por `=`, opcionalmente seguido por espacio en blanco, seguido por `"http://`, seguido por caracteres hasta encontrar `"`, opcionalmente seguido por espacio en blanco, entonces un `>`.
``
- Realiza los ejercicios de <https://alf.nu/RegexGolf>
- Crea una expresión para detectar la palabra “si”. No debe haber coincidencias parciales.

Paréntesis y agrupamiento

- Al colocar parte de una expresión regular dentro de paréntesis, puede agrupar esa parte de la expresión regular.

Perezoso

`/Perez(oso)/`

Perezoso
Group 1
Full match

`/((Perez)(oso))/`

Perezoso
Group 1 Group 2
Full match

Paréntesis y agrupamiento

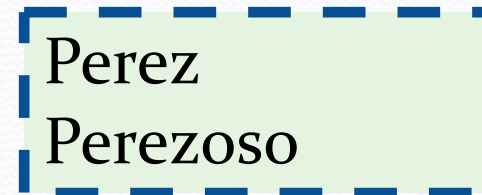
- Los paréntesis también nos ayudan a agrupar sub patrones
- Esto le permite aplicar un cuantificador a todo el grupo o restringir la alternancia a una parte de la expresión regular.

? Significa cero o una vez

`/Perez(oso)?/`

El patrón busca cinco letras: **Perez**

Después cero o una vez las tres letras en el grupo **(oso)?**,



A light green rectangular box with a dashed blue border contains the text 'Perez' on the top line and 'Perezoso' on the bottom line. This illustrates that the regex matches both the base word and the word with the suffix.



A dashed blue rectangular box contains the text 'Otra forma?'. This likely refers to an alternative regex pattern for matching the same set of words.

`/P[eé]rez(oso)?/`

Ejemplo

Nombres de archivo con cierto formato

IMG01.jpg

IMG02.jpg

IMG03.jpg

¿Cómo coincidir con todo el nombre del archivo (con extensión)?

`IMG[\d{1,2}].jpg`

¿Algún problema?

La cadena de clase está equivocada

`IMG\d{1,2}.jpg`

¿y ahora?

¿Y cómo me quedo solo con el nombre?

`(IMG\d{1,2}).jpg`

Consiguiendo el grupo 1 de cada coincidencia

REGULAR EXPRESSION 3 matches, 33 steps (~0ms)

`/ [IMG\d{1,2}].jpg / gm`

TEST STRING

IMG01.jpg
IMG02.jpg
IMG03.jpg

EXPLANATION

- ▼ `/ [IMG\d{1,2}].jpg / gm`
 - ▼ **1st Capturing Group** `[IMG\d{1,2}]`
 - IMG matches the characters `IMG` literally
 - ▼ `\d{1,2}` matches a digit (equal to `[0-9]`)
 - Quantifier** — Matches between 1 and 2 times, giving back as needed (greedy)

MATCH INFORMATION

Match 1

Full match	0-9	IMG01.jpg
Group 1.	0-5	IMG01

¿También puedo recorrer grupos con python?

Claro, revisar

Ejemplos

Si tenemos `/\w*\d+/`

El patrón busca cero o mas caracteres alfanuméricos: `\w*`

Después uno o mas dígitos `\d+`,

Si tenemos la cadena
The12thRobotIs2ndInLine

¿Qué parte coincide con el patrón?

Repeticiones

- Greedy vs Nongreedy matching (Emparejamiento codicioso y no codicioso)
 - Greedy matching obtiene el resultado mas largo posible
 - Nongreedy matching obtiene el mas corto posible
- Digamos que 'The12thRobotIs2ndInLine'
- `/\w*\d+/` (greedy)
 - Coincide con The12thRobotIs2
 - Maximiza la longitud de `\w`
- `/\w*?\d+/` (nongreedy)
 - Coincide con The12
 - Minimiza la longitud de `\w`

Greedy vs Nongreedy matching

- Suponer la cadena 'something is so cool'
 - `/something/`
 - Coincide con 'something'
 - `/so(mething)?/`
 - Coincide con 'something' y el segundo 'so'
 - `/so(mething)??/`
 - Coincide con 'so' y el segundo 'so'
 - No tiene sentido hacer esto

Alternación – múltiples posibilidades

- Sea la cadena ‘He went to get his mother’
 - `/^(He|She)\b.*?\b(his|her)\b.*? (mother|father|brother|sister|dog)/`
 - También coincide con ‘She punched her fat brother’
- Asegurarse que el agrupamiento es correcto!
 - `/^(true|false)$/`
 - Coincide solamente ‘true’ o ‘false’
 - `/^true|false$/` (igual a `/(^true|false$)/`)
 - Coincide con ‘true never’ o ‘not really false’

Guía rápida de expresiones regulares

- **^** Coincide con el **principio** de una línea
- **\$** Coincide con el **fin** de una línea
- **.** Coincide **Cualquier** caracter
- **\s** Coincide (Matches) **whitespace**
- **\S** Coincide cualquier caracter no en blanco (**non-whitespace**)
- ***** **Repite** un caracter cero o mas veces (cuantificador)
- ***?** **Repite** un caracter cero o mas veces (non-greedy)
- **+** **Repite** un caracter una o mas veces (cuantificador)
- **+?** **Repite** un caracter cero o mas veces (non-greedy)
- **[aeiou]** Coincide un solo caracter en el **conjunto** definido
- **|** OR
- **\w** caracter alfanumérico
- **\d** dígito [0-9]
- **\D** caracter que no sea dígito
- **{n,m}** se repite de n a m veces el elemento a su izquierda

Look ahead, Look behind

- Especifica un patrón que se antecede o sucede por cierto grupo o patrón
 - No se capturan dichos grupos

Si tenemos `/(?<=super)man/`

El patrón busca la cadena man solo si esta después de super

No se captura super,

Look ahead, Look behind

- Especifica un patrón que se antecede o sucede por cierto grupo o patrón
 - No se capturan dichos grupos

Si tenemos `/super(?=man)/`

El patrón busca la cadena super solo si esta antes que man

No se captura man,

Ejercicio

- Quiero una expresión para el CURP
 - “AACM651123MTSLLRo6, SAPM880429MTSNRR00, ROBG900321MTSDRD01”
- Quiero solo la fecha de nacimiento sacada del curp
- Expresión que coincida con Juan Pérez o con Juan Alberto Pérez
 - “Juan Alberto Pérez se conoce como Juan Pérez”
- Coincidir con direcciones IP v4
 - “El servidor tiene dirección 10.18.0.20 y la impresora la dirección 10.18.0.116.”
- Cantidades mayores a 4000
 - “El precio de este juguete es 4000 pero aquél cuesta 4010”

Ejercicio

- Quiero coincidencias de etiquetas `<h1>` (cabeceras)
 - `<h1>Esta es una cabecera.</h1><h1>Esta es otra.</h1>`
- Queremos coincidir con caracteres USD solo si están seguidos de una cantidad numérica
 - USD 100
 - USD 350
 - USD 12,345
- Pero no
 - USD currency
 - USD rate

Ejercicio

- Tengo una lista y quiero

- | S.no | Vehicle | Status |
|------|---------|----------|
| 1 | car | sold |
| 2 | car | not sold |
| 3 | car | sold |
| 4 | truck | Repair |

- Quiero los registros de car con Status sold
- Quiero todos los que sean Status sold
- Quiero los que no sean Repair

Ejercicios

- <https://regex.sketchengine.co.uk/>

<https://regexone.com/>