



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Teoría Computacional

Practica 2:Expresiones Regulares

Alumna: Ramírez Galindo Karina

Profesor:Rosas Trigueros Jorge Luis

Fecha de realización:09-09-2019

Fecha de entrega: 23-09-2019



Marco Teórico

Expresiones regulares

Una de las tareas más utilizadas en la programación es la búsqueda de subcadenas o patrones dentro de otras cadenas de texto. [1]

Las expresiones regulares, también conocidas como 'regex' o 'regexp', son patrones de búsqueda definidos con una sintaxis formal. Siempre que sigamos sus reglas, podremos realizar búsquedas simples y avanzadas, que utilizadas en conjunto con otras funcionalidades, las vuelven una de las opciones más útiles e importantes de cualquier lenguaje. [2]

Componentes de las expresiones regulares

Literales: Cualquier caracter se encuentra a sí mismo, a menos que se trate de un metacaracter con significado especial.

Secuencias de escape: La sintaxis de las expresiones regulares nos permite utilizar las secuencias de escape que ya conocemos de otros lenguajes de programación para esos casos especiales como ser finales de línea, tabs, barras diagonales, etc.

Grep con expresiones regulares

Una de las utilidades poco explotadas del grep es el uso de expresiones regulares. [3]

Algunos de sus parámetros para construir las expresiones regulares son: [4]

^ Indica inicio de línea

\$ Indica final de línea

\ Evita dar significado especial a un carácter

^ haríamos: \^

[] Indica un conjunto de caracteres o un rango

[^] Indica la negación de un conjunto de caracteres

. (Punto) Indica cualquier carácter

* (Asterisco) Indica que puede existir el carácter o expresión anterior ninguna o varias veces

\{x,z\} Indica que el carácter o expresión anterior puede existir entre x y z veces

\{x\} Indica que el carácter o expresión debe repetirse exactamente x veces

\{x,\} Indica que el carácter o expresión debe repetirse x veces o más

Direcciones IPv4

El Protocolo de Internet versión 4 en inglés, Internet Protocol version 4 (IPv4), es la cuarta versión del Internet Protocol (IP), un protocolo de interconexión de redes basados en Internet, y fue la primera versión implementada para la producción de ARPANET, en 1983. Definida en el RFC 791. IPv4 usa direcciones de 32 bits, limitándola a

$$2^{32} = 4294967296$$

Direcciones únicas, muchas de las cuales están dedicadas a redes locales (LAN). [5]

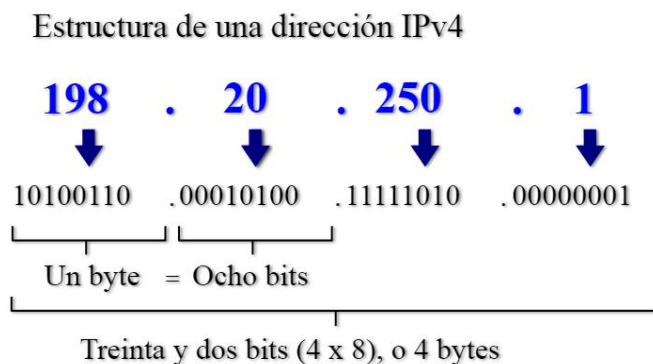


Imagen 1

Estructura IPv4

🚦 Clasificación de direcciones IPv4

Las direcciones IPv4 se clasifican en clases, de ello depende cuantos bits se utilicen para red y cuantos bits se utilicen para host. [5]

Clase de dirección	Rango del primer octeto (decimal)	Bits del primer octeto (los bits verdes no se modifican)	Partes de las direcciones de red (N) y de host (H)	Máscara de subred predeterminada (decimal y binaria)	Cantidad de posibles redes y hosts por red
A	1-127**	00000000-01111111	N.H.H.H	255.0.0.0	128 redes (2^7) 16777214 hosts por red ($2^{24}-2$)
B	128-191	10000000-10111111	N.N.H.H	255.255.0.0	16384 redes (2^{14}) 65534 hosts por red ($2^{16}-2$)
C	192-223	11000000-11011111	N.N.N.H	255.255.255.0	2097150 redes ($2^{21}-2$) 254 hosts por red (2^8-2)
D	224-239	11100000-11101111	NA (multicast)		
E	240-255	11110000-11111111	NA (experimental)		

Imagen 2

Clases de IPv4

🚦 RFC

RFC son las siglas de Registro Federal de Contribuyentes, una clave única que el gobierno mexicano utiliza para identificar a las personas físicas (asalariados) y morales (empresas) que lleven a cabo una actividad económica en nuestro país. Por medio de esta clave la autoridad fiscal es capaz de conocer puntualmente la actividad económica que cada contribuyente lleva a cabo y con quién. [6]



Imagen 3

Formato RFC

Material y equipo

- ✚ Sistema Operativo Ubuntu
- ✚ Python 3
- ✚ Editor de texto

Desarrollo de la práctica

Objetivo:

Desarrollar expresiones regulares (regrexp's) en grep y Python.

Ejercicios:

- ✚ Ingresa una expresión regular que coincida con todos los elementos en la primera columna (ejemplos positivos) pero ninguno de los de la segunda (ejemplos negativos)
- 1) *Cadenas que empiecen con cualquier carácter repetido cero o más veces, seguido de una letra "p", después cualquier carácter, seguido de un letra "t" y después cualquier carácter con cero o más repeticiones.*

Exercise 1

Enter a regexp that matches all the items in the first column (positive examples)

Regexp:

Positive	Negative
pit	pt
spot	Pot
spate	peat
slap two	part
respite	

Imagen 4

Expresión Regular 1

- 2) Cadenas que empiecen con cualquier carácter con cero o más repeticiones, seguido de las letras “ap”, después cualquier carácter con cero o más repeticiones, seguido de una letra “t”, finalmente cualquiera carácter con cero o más repeticiones

Exercise 2

Enter a regexp that matches all the items in the first column (positive examples)

Regexp:

Positive	Negative
	aleht
rap them	happy them
tapeth	tarpth
apth	Apt
wrap/try	peth
sap tray	tarreth
87ap9th	ddapdg
apothecary	apples
	shape the

Imagen 5

Expresión Regular 2

- 3) Cadenas que empiecen con cualquier carácter con cero o más repeticiones, seguido de una letra “f”, después cualquier carácter con cero o más repeticiones, seguido de una letra “k”, después cualquier carácter con cero o más repeticiones, luego que no contengan la letra “o”, o cadenas que empiecen con la letra “b”, seguidas de cualquier carácter con cero o más repeticiones.

Exercise 3

Enter a regexp that matches all the items in the first column (positive examples)

Regexp:

Positive	Negative
affgfking	fgok
rafgkahe	a fgk
bafghk	affgm
baffgkit	afffhk
affgfking	fgok
rafgkahe	afg.K
bafghk	aff gm
baffg kit	afffhgk

Imagen 6

Expresión Regular 3

Encontrar dónde termina una oración y comienza otra es más complicado de lo que podría imaginarse. Ingrese una expresión regular que coincida con todos los elementos en la primera columna (ejemplos positivos) pero ninguno de los de la segunda (ejemplos negativos).

- 4) Cadenas que empiecen con cualquier carácter con cero o más repeticiones, seguido de las letras "ss", después lo que sea. O cadenas que empiecen con cualquier carácter con cero o más repeticiones, seguida de las letras "wa" y después lo que sea. O cadenas que empiecen con cualquier carácter con cero o más repeticiones, seguido de las letras "rr" y después lo que sea. O por último, cadenas que empiecen con las letras "do" y después tengan cualquier carácter.

Exercise 4: Finding sentence breaks

Finding where one sentence ends and another begins is trickier than might be imagined. (see the following examples). When you press "submit", you will see what matched.

Regexp:

Positive	Negative
assumes word senses. Within	in the U.S.A., people often
does the clustering. In the	John?", he often thought, but
but when? It was hard to tell	weighed 17.5 grams
he arrive." After she had	well ... they'd better not
mess! He did not let it	A.I. has long been a very
it wasn't hers!" She replied	like that", he thought
always thought so.) Then	but W. G. Grace never had much

Imagen 7

Expresión Regular 4

- Desarrollar regexp's para grep y para Python que reconozca: Direcciones IPv4 y RFC's.

1) IPv4

Tomando en cuenta la investigación presentada en el marco teórico, la expresión regular para validar IP's queda de la siguiente forma:

```

Open  [icon] expresiones.py
~/Documents

import re
# Validando una dirección IP
cadena=input("Ingrese una dirección IP: ")
patron = ('^(?:25[0-5]|2[0-4][0-9]|'
          '[01]?[0-9][0-9]?)\.'
          '[01]?[0-9][0-9]?)$')

ip = re.compile(patron)

ip.search(cadena)

print(ip.search(cadena))

```

Imagen 8

Código de regex para reconocer IPv4

El patrón especifica que la dirección debe empezar con alguna de las siguientes especificaciones:

- Que empiece con “25” y después tenga un número del 0 al 5, esto acepta: **250,251,252,253,254,255**
- Que empiece con “2” y después tenga un número del 0 al 4 y un número del 0 al 9, esto acepta: **200,201,202,203,204,205,206,207,208,209,210,211,etc.**
- Que empiecen con “01” y después tengan un dos números del 0 al 9, esto acepta: **199, 192,190, 188, etc.**

Lo anterior debe repetirse 3 veces, posteriormente se indica la terminación de la cadena:

- Que termine con “25” y después tenga un número del 0 al 5, esto acepta: **250,251,252,253,254,255**
- Que termine con “2” y después tenga un número del 0 al 4 y un número del 0 al 9, esto acepta: **200,201,202,203,204,205,206,207,208,209,210,211,etc.**
- Que termine con “01” y después tengan un dos números del 0 al 9, esto acepta: **199, 192,190, 188, etc.**

```
ubuntu@ubuntu:~/Documents$ python3 expresiones.py
Ingrese una direccion IP: 255.255.255.0
<re.Match object; span=(0, 13), match='255.255.255.0'>
ubuntu@ubuntu:~/Documents$ python3 expresiones.py
Ingrese una direccion IP: 172.25.12.52
<re.Match object; span=(0, 12), match='172.25.12.52'>
ubuntu@ubuntu:~/Documents$ python3 expresiones.py
Ingrese una direccion IP: 209.165.202.159
<re.Match object; span=(0, 15), match='209.165.202.159'>
```

Imagen 9

Compilación del código para reconocer IPv4

Vemos en la imagen que acepta las IP's ya que están dentro del rango IPv4.

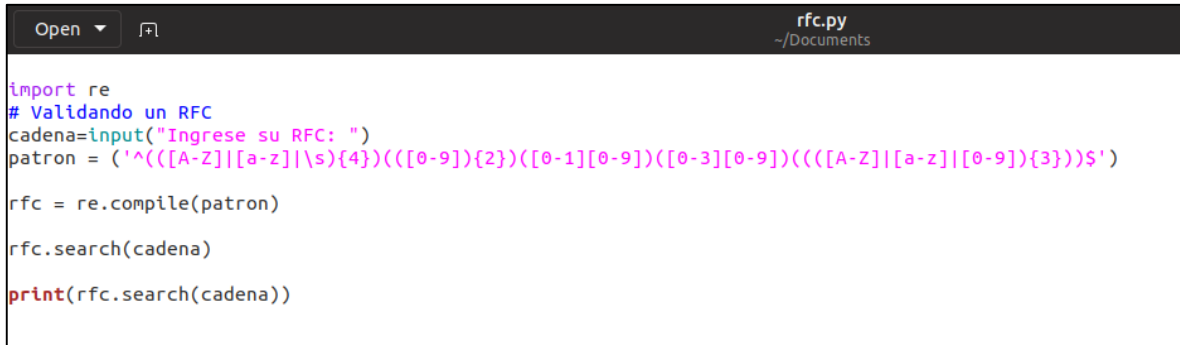
Por otro lado, las siguientes direcciones no las acepta debido a que el límite de los primeros 3 dígitos es 255 y estas sobrepasan dicho límite.

```
ubuntu@ubuntu:~/Documents$ python3 expresiones.py
Ingrese una direccion IP: 258.255.255.0
None
ubuntu@ubuntu:~/Documents$ python3 expresiones.py
Ingrese una direccion IP: 256.147.208.23
None
ubuntu@ubuntu:~/Documents$
```

Imagen 10

2) RFC

La expresión regular para validar un RFC de una persona Física, queda de la siguiente manera



```
import re
# Validando un RFC
cadena=input("Ingrese su RFC: ")
patron = ('^([A-Z][a-z]){4}([0-9]){2}([0-1][0-9])([0-3][0-9])((([A-Z][a-z]){0-9}){3})$')

rfc = re.compile(patron)

rfc.search(cadena)

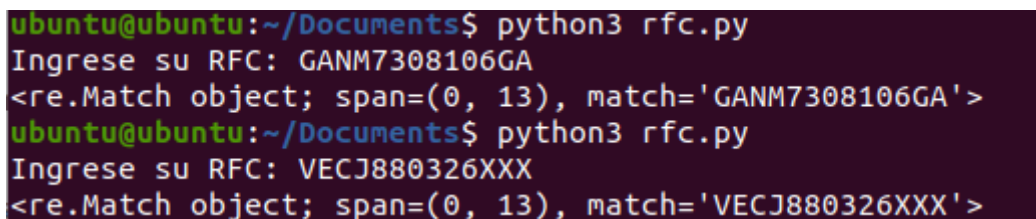
print(rfc.search(cadena))
```

Imagen 11

Código para reconocer RFC

El RFC consta de 13 caracteres, por lo cual el patrón especifica que:

- La cadena debe empezar con letras del abecedario, las reconoce en mayúsculas y minúsculas y especifica que sean 4 caracteres, estos serán los 4 primeros dígitos del RFC.
- Después la cadena tiene que tener dos números entre el 0 y el 9, estos son los siguientes dos dígitos del RFC que indican el año de nacimiento.
- Luego la cadena debe tener un número entre 0 y 1 y un número entre 0 y 9, los cuales indican el mes de nacimiento, esta especificación puede validar solo meses que existan, por ejemplo: **03, 12, 11, etc.**
- Después la cadena debe tener un número entre 0 y 3 y un número entre 0 y 9, esto para aceptar días válidos, por ejemplo: **01, 03, 12, 24, etc.**
- Finalmente los últimos 3 dígitos se especifica que pueden ser números del 0 al 9 o cualquier letra del abecedario.

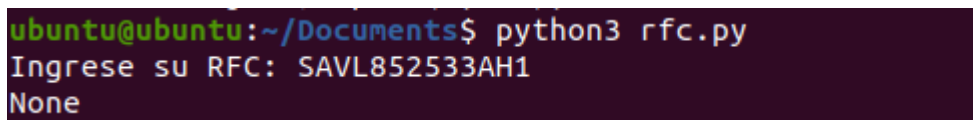


```
ubuntu@ubuntu:~/Documents$ python3 rfc.py
Ingrese su RFC: GANM7308106GA
<re.Match object; span=(0, 13), match='GANM7308106GA'>
ubuntu@ubuntu:~/Documents$ python3 rfc.py
Ingrese su RFC: VECJ880326XXX
<re.Match object; span=(0, 13), match='VECJ880326XXX'>
```

Imagen 12

Compilación del código que reconoce RFC

Las siguientes cadenas no las acepta debido a que el mes “25” no es valido



```
ubuntu@ubuntu:~/Documents$ python3 rfc.py
Ingrese su RFC: SAVL852533AH1
None
```


Conclusiones

Con esta práctica aprendí mas acerca de las expresiones regulares, así como su utilidad tal es el caso del reconocimiento de patrones como se vio en los ejercicios anteriores.

Hacer que la expresión validara solo lo necesario para reconocer las IP's o los RFC's me pareció algo muy interesante ya que en la programación se utiliza mucho este concepto en los motores de búsqueda que permiten automatizar el proceso de búsqueda de modo que sea posible utilizarlo muchas veces para un propósito específico

Referencias

- [1] S. Platzi, «Guía de expresiones regulares en Python,» [En línea]. Available: <https://platzi.com/blog/expresiones-regulares-python/>.
- [2] «Regular-Expressions.info,» [En línea]. Available: <https://www.regular-expressions.info/>.
- [3] [En línea]. Available: <http://systemadmin.es/2009/10/grep-con-expresiones-regulares>.
- [4] «Cómo buscar patrones con grep,» [En línea]. Available: <https://docs.oracle.com/cd/E19620-01/805-7644/6j76klop3/index.html>.
- [5] C. Systems, «Aspectos básicos de networking: Capítulo 6,» 2007. [En línea]. Available: https://moodle2.unid.edu.mx/dts_cursos_mdI/lic/TI/BN/AM/05/Direccionamiento_de_red_IPV4.pdf.
- [6] CONDUCEF, «Registro Federal de Contribuyentes,» [En línea]. Available: <https://www.condusef.gob.mx/Revista/index.php/usuario-inteligente/servicios-financieros/392-registro-federal-de-contribuyentes>.