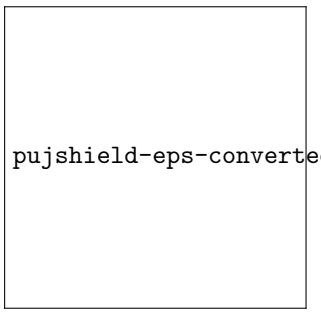

Introduction to regular expressions

Notes of class

Iván Andrés Trujillo Abella

Facultad de ingeniería

---

1 Freddy investigation

ASCII ? UNICODE?

2 Regular expressions

Find patterns in a string (search pattern).
are related with wildcards

2.1 Quantifiers

How many times can occur

```
\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b
```

2.2 special meaning

Metacharacteres

- \
- .
- ^
- |
- ?
- *
- +
- ()
- []

override the special meaning with \.
important thing: windows end files with \r\n and unix \n.
[] match only a single iput $a, b, ..$ for instance:

```
ca[tr]^
```

could math car or cat..
whith - we list a sequence of characteres.

```
[A-Z]
```

we can uses ^to avoid inside the square bracket the character, for instance:

```
le[^o]n
```

```
leon
len
```

2.3 .

Match any character except a new line \n.

2.4 position

^, $
^at the start the string
$ to the end of string

2.4.1 word boundary

\b is a combination of and ^$. boundary word.
we can uses the boundary to find full words, what happen when the word finis with (!,?, - or whit space)?.

```
regex = \bwords\b
words compound language
sentences are composesd of words_
words?
words"
```

B find when the patter no it is and start or end the word.

2.5 consider

```
\w  = [A-Za-z0-9_]
\d = [0-9]
\D = [^0-9]
\s = any space, tab or newline.
```

2.6 or

```
regex = (ML|deep learning) techniques
ML deep learning techniques
ML techniques
deep learning techniques
ML is a technique
```

```
regex = ML|deep learning techniques
in a text could be ML techniques
or deep learning techniques becuase
```

3 repetition

? indicating optional, match zero or more.

```
regex = "?optional
optional\n)
"optional
```

* zero or more times + once or more { int } the number of repetitions allowed. match only number of four digits:

```
regex = \b[0-9]{4}\b
1234
4566
89091
```

```
regex = \b[1-9][0-9]{int,int}\b
```

integers define the range of limitations.

```
regex = \b[a-z]{2,5}\b
last
focus on simple
```

We can say that the number of repetitions are in the last code we can say that { at least , at most }. if 'at most' not is defined then assume {at least , } that the uppermost boundary is indefinite.

4 take in mind

*,+,? are greedy, they match all text.

inside the square brackets not is neccesary escape quantifiers.

? symbol could be used to the end of a quantifier to make a not greedy search. also could be used with { }, in this case match only few as possible.

```
regex = carto{1,n}?
cartooon
```

in a general case only match { at least

```
regex = 'c.*n'
regex = 'c.*?n
```

5 ()

start and the end of a group

6 Another examples

```
regex = cart(o){1,}n
cartoon
cartooooooon
```

7 |

not is greedy.

8 []

the square brackets is a set.

```
'[A-Z][a-z]+', 'one uppercase followed by lowercase'
```

9 Excercises

match the vowels.

10 search and findall methods in python

```
import re
re.match(regex, string)
```

```
re.search(regex, string) # the first location
re.findall(regex, string) #return an array
```

```
string = " Here it is important recognize some emails for instance IA@gmail.com or maybe
    support@somemail.es"
re.findall('[a-z]+@[a-z]+\.[a-z]*',texto)
```

10.1 avoid

The symbol caret allow us not search a set of characters.

`[^abc]`

- confidence interval
- difference paired data
- test population proportion
- comparing two independent means https://www.coursera.org/learn/inferential-statistical-analysis-pythonsyllabus

replace letters by empty.

```
re.sub(regex, to_replace, count)
#count how many replacements will be
```

Could be useful the following code:

```
import re

def repl_func(match):
  if match == True:
    return " "

string = "Hello! How are you?! Where have you been?!"
new_string = re.sub(r"""[!?'".<>(){}@%&*/[/]""", repl_func, string)
print(new_string)
```

11 Fuzzy string matching

if a, b are string the edit distance is the number of operations of edition to transform a in b.

We could think in the following operations over $\xi(emptystringandy)$

insertion

deletion

substitution

11.1 Edit distance

11.2 Background

Similarity of DNA sequences: