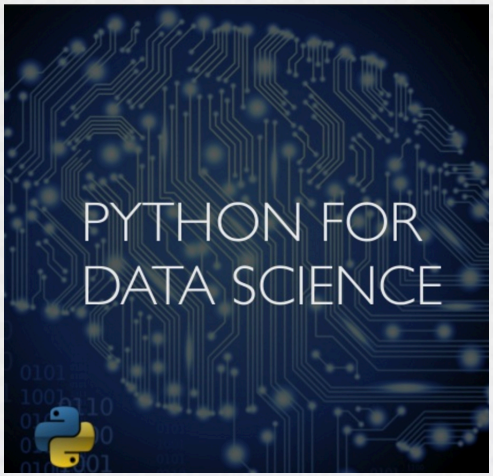
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS



Python for Data Science

Lecture 15 (05/08, 05/10): Text Mining

Decision, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2017

Practical examples

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Matching a password
  - 6 to 12 characters in length
  - Must have at least one uppercase letter
  - Must have at least one lower case letter
  - Must have at least one digit
  - Should contain other characters

Practical examples

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Matching URL
  - Must start with http or https or ftp followed by ://
  - Must match a valid domain name
  - Could contain a port specification (http://www.sitepoint.com:80)
  - Could contain digit, letter, dots, hyphens, forward slashes, multiple times

Practical examples

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Matching duplicated words
  - The words are space separated
  - We must match every duplication – non-consecutive ones as well

Regular expression

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- In computing, a regular expression, also referred to as “regex” or “regexp”, provides a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters.

Regular expression

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Use regular expressions to:
  - ❑Search a string (search and match)
  - ❑Replace parts of a string (sub)
  - ❑Break strings into smaller pieces (split)

Regular expression syntax

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Most characters match themselves
  - The regular expression "test" matches the string 'test', and only that string
- [x] matches any *one* of a list of characters
  - "[abc]" matches 'a', 'b', or 'c'
- [^x] matches any *one* character that is not included in x
  - "[^abc]" matches any single character *except* 'a', 'b', or 'c'

Regular expression syntax

- "." matches any single character
- Parentheses can be used for grouping
  - "(abc)" matches 'abc'
- xly matches *x* or *y*
  - "thislthat" matches 'this' or 'that', but not 'thisthat'.

Regular expression syntax

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- *x** matches zero or more *x*'s
  - "a*" matches '', 'a', 'aa', etc.
- *x*+ matches one or more *x*'s
  - "a+" matches 'a', 'aa', 'aaa', etc.
- *x*? matches zero or one *x*'s
  - "a?" matches '' or 'a'
- *x{m, n}* matches *i* *x*'s, where $m \leq i \leq n$
  - "a{2,3}" matches 'aa' or 'aaa'

Regular expression syntax

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- “\d” matches any digit; “\D” any non-digit
- “\s” matches any whitespace character; “\S” any non-whitespace character
- “\w” matches any alphanumeric character; “\W” any non-alphanumeric character
- “^” matches the beginning of the string; “$” the end of the string
- “\b” matches a word boundary; “\B” matches a character that is not a word boundary

Regular expression module

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Before we use regular expressions in our program, we must import the library using

```
import re
```

Search and match

- The two basic functions are **re.search** and **re.match**
  - Search looks for a pattern anywhere in a string
  - Match looks for a match starting at the beginning
- Both return *None* (logical false) if the pattern isn't found and a "match object" instance if it is

```
>>> import re
>>> pat = "a*b"
>>> re.search(pat,"fooaaabcde")
<_sre.SRE_Match object at 0x809c0>
>>> print re.match(pat,"fooaaabcde")
None
```

Using re.search()

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
import re

fh = open('test.txt','r')
for line in fh:
    line = line.rstrip()
    if re.search('^Hello', line):
        print line
```

test.txt:

Hello world
World, hello
Hello, hello
hello world
World
world, Hello

Output:

Hello world
Hello, hello

Group() function

- Using group to get the matched object

```
>>> r1 = re.search("a*b","fooaaabcde")
>>> r1.group()  # group returns string matched
'aaab'
>>> r1.start()  # index of the match start
3
>>> r1.end()    # index of the match end
7
>>> r1.span()   # tuple of (start, end)
(3, 7)
```

What got matched?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Here's a pattern to match simple email addresses

  \w+@(\w+\.)+(com|org|net|edu)

```
>>> pat1 = "\w+@(\w+\.)+(com|org|net|edu)"
>>> r1 = re.match(pat, "alice@umd.edu")
>>> r1.group()
'alice@umd.edu'
```

- We might want to extract the pattern parts, like the email name and host

What got matched?

- We can put parentheses around groups we want to be able to reference

```
>>> pat2 = "(\w+)@( (\w+\.)+( com | org | net | edu ) )"
>>> r2 = re.match(pat2, "alice@umd.edu")
>>> r2.group(1)
'alice'
>>> r2.group(2)
'umd.edu'
>>> r2.groups()
r2.groups()
('alice', 'umd.edu', 'umd.', 'edu' )
```

- Note that the ‘groups’ are numbered in a preorder traversal of the forest

re.findall()

- When we use re.findall() it returns a list of zero or more sub-strings that match the regular expression

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42’
>>> y = re.findall('[0-9]+', x)
>>> print y
['2', '19', '42']
>>> y = re.findall('[AEIOU]+', x)
>>> print y
[]
```

More re functions

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- re.split() is like split but can use patterns

```
>>> re.split("\W+", “This... is a test,  short and sweet, of split().”)
['This', 'is', 'a', 'test', 'short',  'and', 'sweet', 'of', 'split', '']
```

- re.sub() substitutes one string for a pattern

```
>>> re.sub('(blue | white | red)', 'black', 'blue socks and red shoes')
'black socks and black shoes’
```

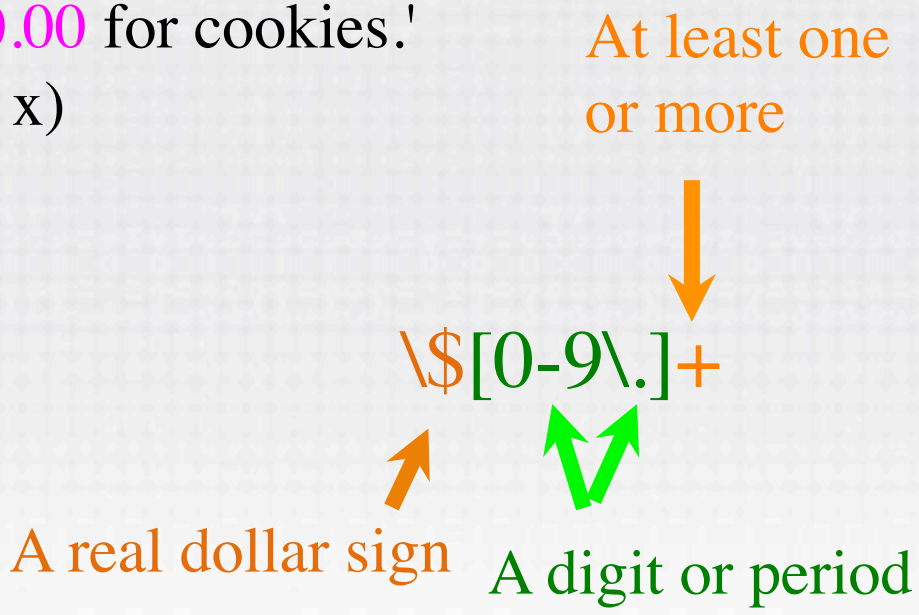
- re.findall() finds all matches

```
>>> re.findall("\d+","12 dogs,11 cats, 1 egg")
['12', '11', '1']
```

Escape Character

- If you want a special regular expression character to just behave normally (most of the time) you prefix it with '\'

```
>>> import re
>>> x = 'We just received $10.00 for cookies.'
>>> y = re.findall('\$[0-9\.]+', x)
>>> print y
['$10.00']
```



Regular expression quick guide

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
^          Matches the beginning of a line
$          Matches the end of the line
.          Matches any character
\s         Matches whitespace
\S         Matches any non-whitespace character
*          Repeats a character zero or more times
*?         Repeats a character zero or more times (non-greedy)
+          Repeats a chracter one or more times
+?         Repeats a character one or more times (non-greedy)
[aeiou]    Matches a single character in the listed set
[^XYZ]     Matches a single character not in the listed set
[a-z0-9]   The set of characters can include a range
(          Indicates where string extraction is to start
)          Indicates where string extraction is to end
```

```
^(http|https|ftp)\:[\/]{2}
([a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,4})
(:[0-9]+)?\/?
([a-zA-Z0-9\-\._\?\,\'\/\\\+&amp;%\$#\=~]*)
```

1. `^` asserts position at start of the string
2. capturing group `(http|https|ftp)`, captures `http` or `https` or `ftp`
3. `:` escaped character, matches the character `:` literally
4. `[\/]{2}` matches exactly 2 times the escaped character `/`
5. capturing group `([a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,4})`:
   - `[a-zA-Z0-9\-\.]+` matches one and unlimited times character in the range between a and z, A and Z, 0 and 9, the character `-` literally and the character `.` literally
   - `\.` matches the character `.` literally
   - `[a-zA-Z]{2,4}` matches a single character between 2 and 4 times between a and z or A and Z (case sensitive)
6. capturing group `(:[0-9]+)?`:
   - quantifier `?` matches the group between zero or more times
   - `:` matches the character `:` literally
   - `[0-9]+` matches a single character between 0 and 9 one or more times
7. `\/?` matches the character `/` literally zero or one time
8. capturing group `([a-zA-Z0-9\-\._\?\,\'\/\\\+&%\$#\=~]*)`:
   - `[a-zA-Z0-9\-\._\?\,\'\/\\\+&%\$#\=~]*` matches between zero and unlimited times a single character in the range a-z, A-Z, 0-9, the characters: `-._?,'/\+&%$#=~`.

Matching duplicated words

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

```
\b(\w+)\b(.*\1)
```