NLP For Economists

Regular Expressions

Sowmya Vajjala

Munich Graduate School of Economics - LMU Munich

Guest Course, October 2020

Goals

- Introduce regular expressions and why they are useful for NLP
- source: Chapter 11 in Py4E book

Regular Expressions

- Regular expressions are used to do do pattern based information extraction from data.
- They have their own syntax for doing pattern matching in different ways.
- They are very useful to process text and manipulate it.
- Regular expressions in python are in a module called "re" and you can use them in your code once you add a "import re" statement in your program/console.
- They can simplify a lot of your tasks, but they themselves can be very complicated.

pythe

Your regular expression:

/ \d{3}

IGNORECASE MULTILINE DOTALL

Your test string:

515-294-5100 Facilities Planning and Management
515-294-4444 Help Van
Poison Control
800-222-1222 Poison Control Center

Match result:

515-294-4428 Department of Public Safety
515-294-3322 Department of Residence
515-294-5100 Facilities Planning and Management
515-294-4444 Help Van
Poison Control
800-222-1222 Poison Control Center

RegEx syntax

1. ^ matches the beginning of a line. For example,
   - a pattern ^Th matches all lines in a text file that start with Th
2. $ matches the end of a line. For example,
   - a pattern Th$ matches all lines in a text file that end with Th
3. \s matches a white space character
4. \S matches a non-white space character.

RegEx syntax

1. . matches any character
2. * -applies to the immediately preceding character and indicates to match zero or more of the preceding character(s).
   - for example, te* matches all locations where there is a t, te, tt, tete etc.
3. + - applies to the immediately preceding character and indicates to match one or more of the preceding character(s).
   - for example, te+ matches all locations where there is a te, tete, tetete etc.

RegEx syntax - continued

The power of square brackets

1. [aeiou]- matches a single character as long as the character is in this set.
2. You can also specify ranges in square brackets. For example, [a-z0-9] matches all characters in lower case or a single digit.
3. When the first character after the square brackets is a caret (^), it works like a "not" keyword. So, [^a-z0-9] matches all characters that are not lower cased letters, and not numbers.

Escape Character

What do you do if you want to match a ? or a . which also carry a meaning in regex?

Escape Character

What do you do if you want to match a ? or a . which also carry a meaning in regex?
We "escape" them to tell regex module that these are real characters and not regex syntax. This is done using a \ character.

So, st\. is a pattern that searches for all occurences of "st." in a string.

Using RegEx in Python

Python's "re" module

- You should import "re" module of python using the statement: import re
- Two useful functions in re module are: search() and findall().
- search() in re module is similar to the find() method for Strings, but just more sophisticated.
  - re.search("XX[0-9]",str) searches for the first occurence of "XX" followed by a digit in a string and returns the corresponding match.
- findall() returns all matches of a pattern in a string, as a list of matches.

Searching a text with RegEx

re.search() function

```
import re
filecontent = open('mbox.txt') #This line opens a file called mbox.txt for reading.
for line in filecontent: #This loop reads a file line by line.
    line = line.rstrip() #What is this doing?

    '''#Example 1: search for a normal string. We can use normal find() method of a string too!
    if re.search('localhost', line):
        print(line) #This prints all lines which has "localhost" somewhere.'''

    '''#Example 2: search for a string that starts with x. We can use startsWith() instead of this!
    if re.search('^X', line):
        print(line) #This prints all lines that start with X'''

    #Example 3: Here is where regular expression search is really useful.
    if re.search('[0-9]+\sDec\s2007',line):
        print(line)
```

Searching a text with RegEx

re.findall() function

```
import re
s = 'Hello from csev@umich.edu to cwen@iupui.edu about the meeting @2PM'
searchmatch = re.search('\S+@\S+', s)
findallmatch = re.findall('\S+@\S+', s) #This is a "LIST" object
#print(searchmatch)
print(findallmatch[1][4])
#print(findallmatch[0])
```

Searching a text with user given Regex

```
#Program to emulate grep. A question from textbook.
import re
hand = open('mbox.txt')
pattern = input("Enter the pattern you want to see: ") #user enters a pattern
count = 0 #variable to save the the number of times a pattern has occured

#Now reading the file line by line
for line in hand:
    if re.search(pattern,line):
        count = count +1

print("mbox.txt had " + str(count) +" lines that matched " + pattern)
```

Concluding Remarks

- I just touched upon very basic utilities of regular expressions
- If used properly, they are a very powerful tool to do pattern driven information extraction
- In domains where usage of NLP methods is relatively new, and we don't have a lot of annotated data, regular expressions can give very useful first solutions!