Regular Expression in Natural Language Processing

Basic Text Processing (Regular Expressions)

- A sequence of symbols and characters expressing a string or pattern to be searched for within a longer piece of text.
- A formal language to specify text string
- Used to accomodate Misspellings
- Example
  - Donation
  - Donated
  - Donating
  - donates

RE --- [Dd]onat (ion|ed|ing|es)

Regular Expression Character class

Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient

For Numbers ?

RE: Negations

- Negations [^Ss]
  - Carat means negation only when first in []

Pattern	Matches	
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"

RE: Wild cards (? * + .)

- **A meta-character is a character that has a special meaning (instead of a literal meaning)**
- **The meta character matches any character is called Wild cards**

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n

RE: Disjunctions

Two atoms or groups separated by the meta character | (vertical bar) indicate the disjunction

- The pipe | for disjunction

Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

Regular Expressions: Anchors ^ $

Anchors do not match any character at all. Instead, they match a position before, after, or between characters

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\.$	The end.
.$	The end? The end!

RE to solve research problem

Identifier	Regular expression
1	\b(I|we)\b.*\b(am|are|will be)\b.*\b(bringing|giving|helping|raising|donating|auctioning)\b
2	\b(I'm)\b.*\b(bringing|giving|helping|raising|donating|auctioning)\b
3	\b(we're)\b.*\b(bringing|giving|helping|raising|donating|auctioning)\b
4	\b(I|we)\b.*\b(will|would like to)\b.*\b(bring|give|help|raise|donate|auction)\b
5	\b(I|we)\b.*\b(will|would like to)\b.*\b(work|volunteer|assist)\b

H. Purohit, C. Castillo, F. Diaz, A. Sheth, and P. Meier, "Emergency relief coordination on social media: Automatically matching resource requests and offers," First Monday, vol. 19, no. 1, Jan 2014

Python Script

```
import sys
import re

infile = "data-science.txt"
filename=raw_input("Enter file to store output of the Regular Expression")
fileptr = open(filename,'w')

def match(text):

    if re.search(r'exqui[^Ss]',text): #1
     return True
    if re.search(r'/bwillb/',text): #2
     return True
    else:
        return False
```

```
def process_file(infile):

    fin = open(infile, "r")

    for line in fin:
        temp = match(line)
        if temp == True :
            fileptr.write(line)
    #end for
# end function


def main():

    process_file(infile)

# end main()


if __name__ == '__main__':
    main()
```

1 Which of the following matches regexp /a(ab)*a/

1) abababa
2) aaba
3) aabbaa
4) aba
5) aabababa

2 Which of the following matches regexp /ab+c?/

1) abc
2) ac
3) abbb
4) bbc

3 Which of the following matches regexp /a.[bc]+/

1) abc
2) abbbbbbbb
3) azc
4) abcbcbcbc
5) ac
6) asccbbbbcbcccc

4 Which of the following matches regexp /abc|xyz/

1) abc
2) xyz
3) abc|xyz

5 Which of the following matches regexp /[a-z]+[\.\?!]/

1) battle!
2) Hot
3) green
4) swamping.
5) jump up.
6) undulate?
7) is.?

6 Which of the following matches regexp /[a-zA-Z]*[^,]=/

1) Butt=
2) BotHEr,=
3) Ample
4) FIdDlE7h=
5) Brittle =
6) Other.=

7 Which of the following matches regexp /[a-z][\.\?!]\s+[A-Z]/
(\s matches any space character)

1) A. B
2) c! d
3) e f
4) g. H
5) i? J
6) k L

8 Which of the following matches regexp /(very)+(fat)?(tall|ugly) man/

1) very fat man
2) fat tall man
3) very very fat ugly man
4) very very very tall man

Answers

1. 2, 5
2. 1, 3
3. 1, 2, 3, 4, 6
4. 1, 2
5. 1, 4, 6
6. 1, 5, 6
7. 4, 5
8. 3, 4

Exercise 1

regexp that matches all the items in the first column (positive examples) but none of those in the second (negative examples).

Positive

pit
spot
spate
slap two
respite

Negative

pt
Pot
peat
part

Answer

1. s?.*p.t.*
2. [rs]?(es|a)?p.t.*
3. [rs]?(es|la)?p.t(e|wo)?

Exercise 2

regexp that matches all the items in the first column (positive examples) but none of those in the second (negative examples).

Positive

rap them
tapeth
apth
wrap/try
sap tray
87ap9th
apothecary

Negative

aleht
tarpth
Apt
peth
tarreth
ddapdg
apples

Answer

.*ap.*t.*

Exercise 3

regexp that matches all the items in the first column (positive examples) but none of those in the second (negative examples).

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

Answer

[bra]a?f?fg[fkh\s].*

Reference

https://regex.sketchengine.co.uk/

Examples

- Find me all instances of the word "the" in a text.

  `the`

  Misses capitalized examples

  `[tT]he`

  Incorrectly returns `other` or `theology`

  `[^a-zA-Z][tT]he[^a-zA-Z]`

ERRORS

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

ERROR Cotd.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations