Computational Linguistics

Lecture 2: Regular Expressions, Text Normalization and Edit Distance

Athanasios Karasimos

Task 1: Find the correct RegEx. Sometimes there are more than one possible solutions.

1.1 Go ahead and try writing a pattern that matches all three rows, it may be as simple as the common letters on each line.

Match	abcdefg
Match	abcde
Match	abc

<u>Solution</u>:

1.2. Try writing a pattern that matches all the digits in the strings below, and notice how your pattern matches anywhere within the string, not just starting at the first character.

Match	abc123xyz
Match	define "123”
Match	var g = 123;

<u>Solution</u>:

1.3 Below are a couple strings with varying characters but the same length. Try to write a single pattern that can match the first three strings, but not the last (to be skipped). You may find that you will have to escape the dot metacharacter to match the period in some of the lines.

Match	cat.
Match	896.
Match	?=+.
Skip	abc1

<u>Solution</u>:

1.4 Below are a couple lines, where we only want to match the first three strings, but not the last three strings. Notice how we can't avoid matching the last three strings if we use the dot, but have to specifically define what letters to match using the notation above.

Match	can
Match	man
Match	fan
Skip	dan
Skip	ran
Skip	pan

Solution:

1.5 With the strings below, try writing a pattern that matches only the live animals (hog, dog, but not bog). Notice how most patterns of this type can also be written using the technique from the last lesson as they are really two sides of the same coin. By having both choices, you can decide which one is easier to write and understand when composing your own patterns.

Match	hog
Match	dog
Skip	bog

Solution:

1.6 In the exercise below, notice how all the match and skip lines have a pattern, and use the bracket notation to match or skip each character from each line. Be aware that patterns are case sensitive and a-z differs from A-Z in terms of the characters it matches (lower vs upper case).

Match	Ana
Match	Bob
Match	Cpc
Skip	aax
Skip	bby
Skip	ccz

Solution:

1.7 In the lines below, the last string with only one z isn't what we would consider a proper spelling of the slang "wazzup?". Try writing a pattern that matches only the first two spellings by using the curly brace notation above.

Match	wazzzzzup
Match	wazzzup
Skip	wazup

Solution:

1.8 Below are a few simple strings that you can match using both the star and plus metacharacters.

Match	aaaabcc
Match	aabbbbc
Match	aacc
Skip	a

Solution:

1.9 In the strings below, notice how the plurality of the word "file" depends on the number of files found. Try writing a pattern that uses the optionality metacharacter to match only the lines where one or more files were found.

Match	1 file found?
Match	2 files found?
Match	24 files found?
Skip	No files found.

1.10 In the strings below, you'll find that the content of each line is indented by some whitespace from the index of the line (the number is a part of the text to match). Try writing a pattern that can match each line regardless of how much whitespace is between the number and the content. Notice that the whitespace characters are just like any other character and the special metacharacters like the star and the plus can be used as well.

Match	1. abc
Match	2. abc
Match	3. abc
Skip	4.abc

<u>Solution</u>:

1.11 Note that this is different than the hat used inside a set of bracket [^...] for excluding characters, which can be confusing when reading regular expressions. Try to match each of the strings below using the special characters.

Match	Mission: successful
Skip	Last Mission: unsuccessful
Skip	Next Mission: successful upon capture of target

<u>Solution</u>:

1.12 Go ahead and try to use this to write a regular expression that matches only the filenames (not including extension) of the PDF files below.

Capture	file_record_transcript.pdf
Capture	file_07241999.pdf
Skip	testfile_fake.pdf.tmp

<u>Solution</u>:

1.13 For the following strings, write an expression that matches and captures both the full date, as well as the year of the date.

Capture Jan 1987
Capture May 1969
Capture Aug 2011

<u>Solution</u>:

1.14 Go ahead and try writing a conditional pattern that matches only the lines with small fuzzy creatures below.

Match	I love cats
Match	I love dogs
Skip	I love logs
Skip	I love cogs

Lesson Notes

abc…	Letters
123…	Digits
\d	Any Digit
\D	Any Non-digit character
.	Any Character
\\.	Period
[abc]	Only a, b, or c
[^abc]	Not a, b, nor c
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
{m}	m Repetitions
{m,n}	m to n Repetitions
*	Zero or more repetitions
+	One or more repetitions
?	Optional character
\s	Any Whitespace
\S	Any Non-whitespace character
^…$	Starts and ends
(…)	Capture Group
(a(bc))	Capture Sub-group
(.*)	Capture all
(abc|def)	Matches abc or def