Natural Language Processing - 2nd Semester
(2024-2025)
1038141

1.3 - Regular Expressions

SAPIENZA
UNIVERSITÀ DI ROMA

Prof. Stefano Faralli
faralli@di.uniroma1.it

Prof. Iacopo Masi
masi@di.uniroma1.it

**credits are reported in the last slide

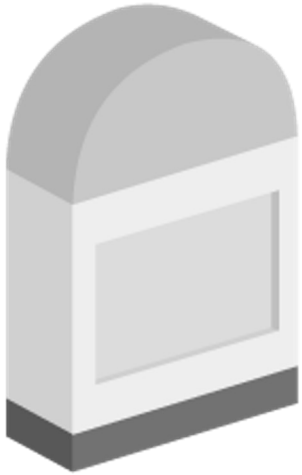
SAPIENZA
UNIVERSITÀ DI ROMA

1.3 - Regular Expressions

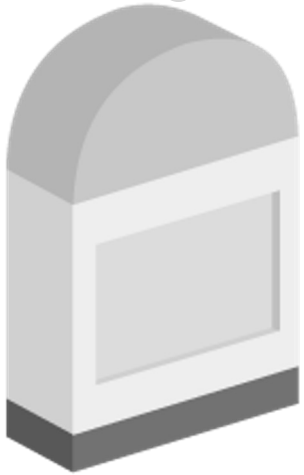
- Tools to test regular expressions
- Milestones
- A real-world scenario
- Regular expressions and pattern matching
- Simple and Complex Regular Expressions
- Regular expressions and NLP: Eliza
- Q&A

Milestones in NLP

rule-based systems



statistical and classical machine learning models



deep learning models

Tools to test regular expressions

Web App
https://regex101.com/

Python
https://www.programiz.com/python-programming/regex

Java
https://www.w3schools.com/java/java_regex.asp

Perl
https://www.geeksforgeeks.org/perl-substitution-operator/

A real-world scenario

- Interested in woodchucks?
- Woodchuck?
- woodchuck?
- Woodchucks?

Regular Expressions and pattern matching

Regular expressions

- a compact textual representation of a set of strings representing a language

Pattern matching

- Everybody does it: Emacs, vi, perl, grep, directory listing, etc.
- Simple but powerful tools for 'shallow' processing, e.g. of very large corpora
  - What word is most likely to begin a sentence?
  - What word is most likely to begin a question?
  - How often do people end sentences with prepositions?
- With other simple statistical tools, allow us to
  - Obtain **word frequency** and **co-occurrence** statistics
    - What is this document 'about'?
    - What words typically *modify* other words? (e.g. politician)
  - Build simple interactive applications (e.g. Eliza)

Some simple regular expressions

- Regular expression search requires a **pattern** that we want to search for, and a **corpus** of texts to search through.

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/!/	"You've left the burglar behind again!" said Nori

[Jurafsky&Martin, 2022]

The use of brackets []

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

RE	Match	Example Patterns Matched
/[A-Z]/	an upper case letter	"we should call it 'Drenched Blossoms' "
/[a-z]/	a lower case letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

[Jurafsky&Martin, 2022]

Caret, Question mark and Period

RE	Match (single characters)	Example Patterns Matched
/[^A-Z]/	not an upper case letter	"Oyfn pripetchik"
/[^Ss]/	neither 'S' nor 's'	"I have no exquisite reason for't"
/[^.]/	not a period	"our resident Djinn"
/[e^]/	either 'e' or '^'	"look up ^ now"
/a^b/	the pattern 'a^b'	"look up a^ b now"

RE	Match	Example Patterns Matched
/woodchucks?/	woodchuck or woodchucks	"woodchuck"
/colou?r/	color or colour	"color"

RE	Match	Example Matches
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

[Jurafsky&Martin, 2022]

Counting

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	*n* occurrences of the previous char or expression
{n,m}	from *n* to *m* occurrences of the previous char or expression
{n,}	at least *n* occurrences of the previous char or expression
{,m}	up to *m* occurrences of the previous char or expression

[Jurafsky&Martin, 2022]

Aliases

RE	Expansion	Match	First Matches
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

RE	Match	First Patterns Matched
*	an asterisk "*"	"K*A*P*L*A*N"
\.	a period "."	"Dr. Livingston, I presume"
\?	a question mark	"Why don't they come and lend a hand?"
\n	a newline	
\t	a tab	

[Jurafsky&Martin, 2022]

Example Exercise

- Find all the instances of the word “the” in a text. https://regex101.com/
  - /the/

[Jurafsky&Martin, 2022]

Example Exercise

- Find all the instances of the word “the” in a text. https://regex101.com/
  - /the/
  - /[tT]he/

Example Exercise

- Find all the instances of the word “the” in a text. https://regex101.com/
  - /the/
  - /[tT]he/
  - /[^a-zA-Z][tT]he[^a-zA-Z]/

Example Exercise

- Find all the instances of the word "the" in a text. https://regex101.com/
  - /the/
  - /[tT]he/
  - /[^a-zA-Z][tT]he[^a-zA-Z]/
  - /(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/

[Jurafsky&Martin, 2022]

Example Exercise

- Find all the instances of the word "the" in a text. https://regex101.com/
  - /the/
  - /[tT]he/
  - /[^a-zA-Z][tT]he[^a-zA-Z]/
  - /(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/
- The process we just went through was based on two fixing kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives
  - Not matching things that we should have matched (The)
    - False negatives
- We'll be telling the same story for many tasks, all semester. 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).

[Jurafsky&Martin, 2022]

Substitutions

- An important use of regular expressions is in **substitutions.**

  The (Perl) substitution operator s/regexp/pattern/ allows a string characterized by a regular expression (regexp) to be replaced by another string (pattern)

  Example:

```
#!/usr/bin/perl -w

# String in which text is to be replaced

$string = "Hello all!!! Welcome here";

# Use of s operator to replace text with pattern

$string =~ s/here/to Geeks/;

# Printing the updated string

print "$string\n";
```

```
GeekstoGeeks
```

https://www.geeksforgeeks.org/perl-substitution-operator/

Registers

- It is often useful to be able to refer to a particular subpart of the string matching: to do this, we put parentheses around parts of the pattern, and use the **number operator** to refer back
- Example:

  regex: /the (.*)er they were, the \1er they will be/min

  text: the researcher they were, the researcher they will be

  text: the researcher they were, the miner they will be

  https://regex101.com/

[Jurafsky&Martin, 2022]

Regular Expressions and NLP: Eliza

- ELIZA is a computer program devised by Joseph Weizenbaum (1966) that simulates the role of a Rogerian psychologist.
- One of the first programs developed that explored the issues involved in using natural language as the mode of communication between humans and the machine.
- Using *simple pattern matching*, without any deeper knowledge of the world or of the conversation.

- https://www.youtube.com/watch?v=4sngIh0YJtk

[Jurafsky&Martin, 2022]

Resources and References

[Jurafsky&Martin, 2022] Jurafsky and Martin. Speech and Language Processing, Prentice Hall, third edition
https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

Regular Expressions Online
https://regex101.com/

Perl Substitution operator
https://www.geeksforgeeks.org/perl-substitution-operator/

Q&A

SAPIENZA
UNIVERSITÀ DI ROMA

Let's revise the available notebook:

https://github.com/iacopomasi/NLP/blob/main/notebooks/Part_1_3_Regular_Expressions.ipynb

**Credits

The slides of this part of the course are the result of a personal reworking of the slides and of the course material from different sources:

1. The NLP course of Prof. Roberto Navigli, Sapienza University of Rome
2. The NLP course of Prof. Simone Paolo Ponzetto, University of Mannheim, Germany
3. The NLP course of Prof. Chris Biemann, University of Hamburg, Germany
4. The NLP course of Prof. Dan Jurafsky, Stanford University, USA