Foundations of NLP

CS3126

Lecture-1

1.1 Regular Expressions

Mahindra™ University

Global Thinkers. Engaged Leaders.

For those who did not join Slack

Acknowledgments

These slides were adapted from the book

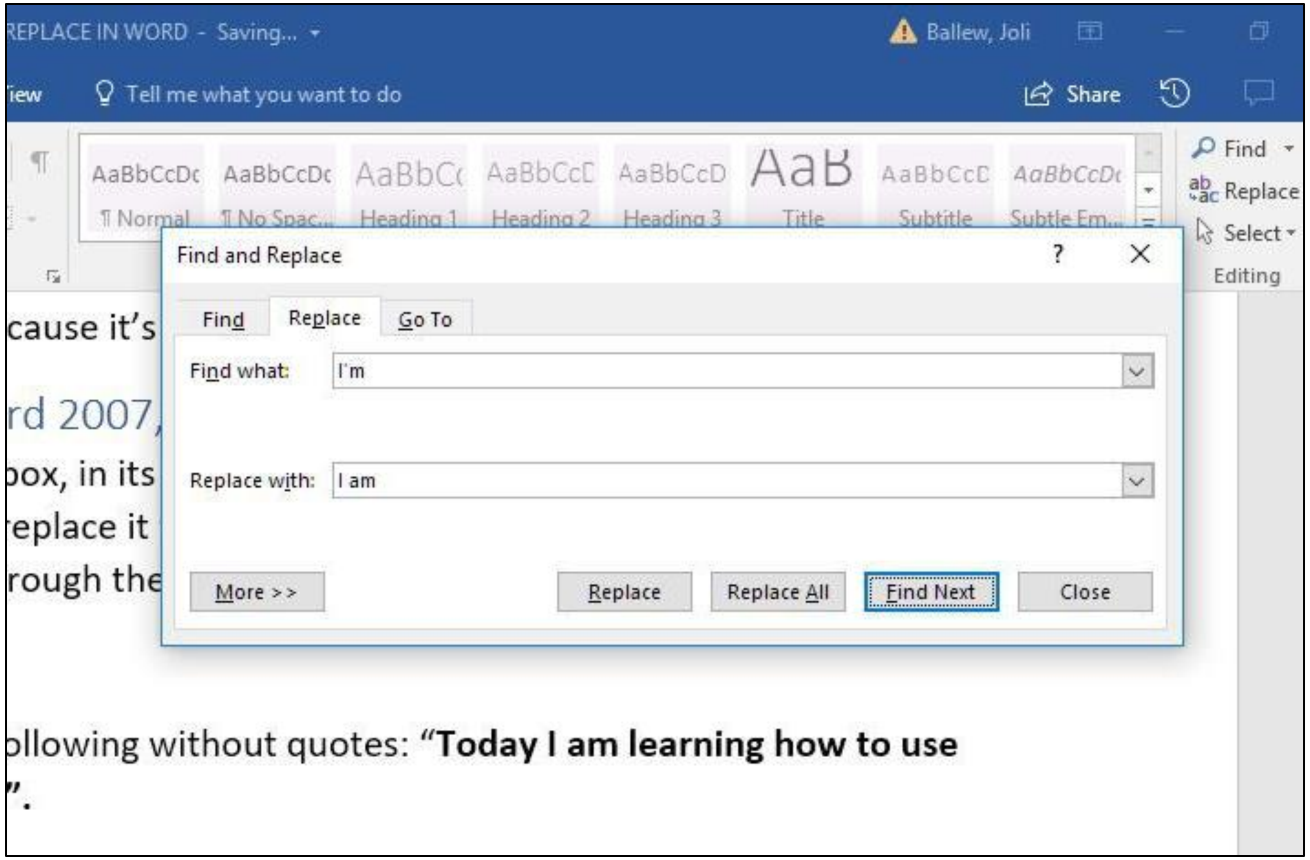
SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition and

Some modifications from presentations and resources found in the WEB by several scholars.

Regular Expressions

- Regular expressions, are sequences of characters that define a search pattern.

- They are used for matching and manipulating text strings based on patterns.



Find what: I'm

Replace with: I am

cause it's

rd 2007,

box, in its

eplace it

rough the

ollowing without quotes: “**Today I am learning how to use**

”.

Where to use Regex?

- Data pre-processing
- Rule-based information mining systems
- Pattern matching
- Text feature engineering
- Web scraping
- Data extraction

many more………

Why?

- Lot of unstructured data
- 1st step is pre-processing
  - Ways to do text pre-processing
    - Regex is one of the tool

Regular Expressions

- Disjunction
  - Negation
  - Pipe |
  - Special characters ? * + .
  - Anchors ^ $

Regular Expressions: Disjunctions

Letters inside square brackets

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

Ranges

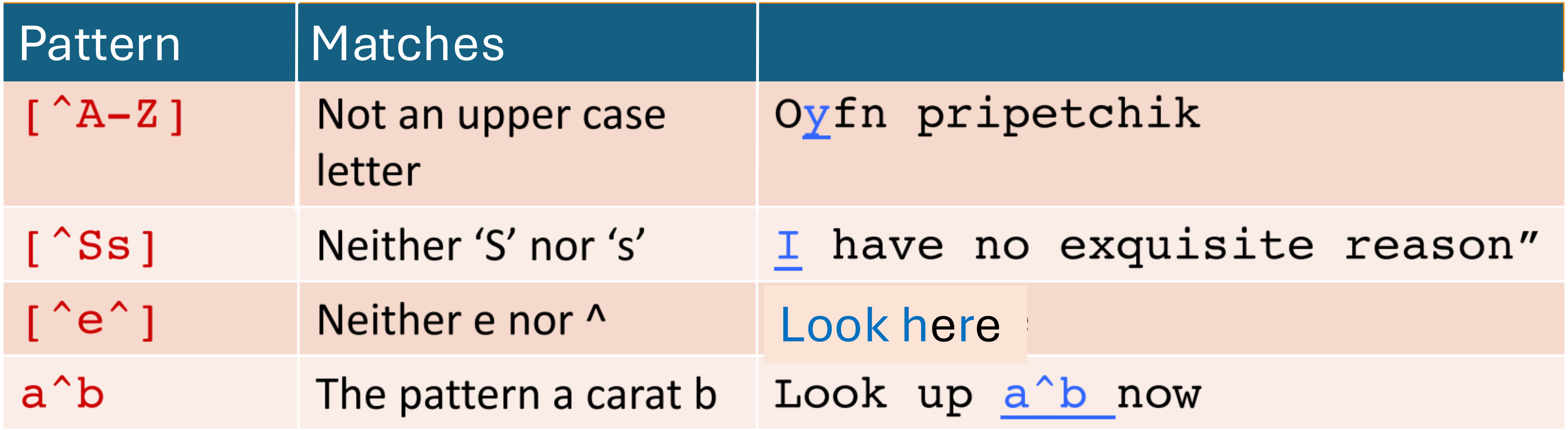
Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

Slide Reference: https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc_2023.pdf

Regular Expressions: Negation in Disjunction

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

Special characters (., *, +, ?) lose their special meaning inside []



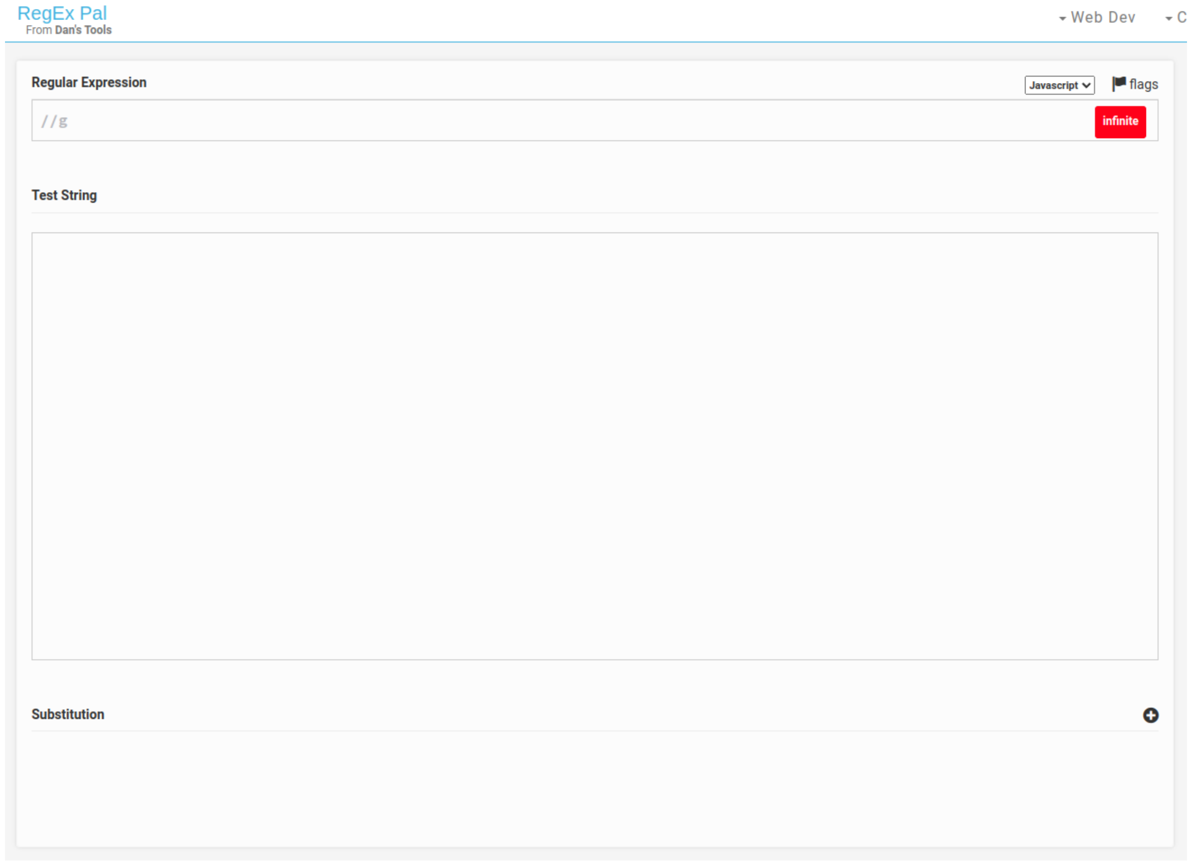
Pattern	Matches	
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[^e^]	Neither e nor ^	Look here
a^b	The pattern a carat b	Look up a^b now

Slide Reference: https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc_2023.pdf

Try Demo tool!!!

Match the patterns such as

- [Ww]
- [A-Z]
- [a-z]
- [A-Za-z]



[Regex Tester - Javascript, PCRE, PHP (regexpal.com)](https://www.regexpal.com)

Regex documentation/Python

Documentation

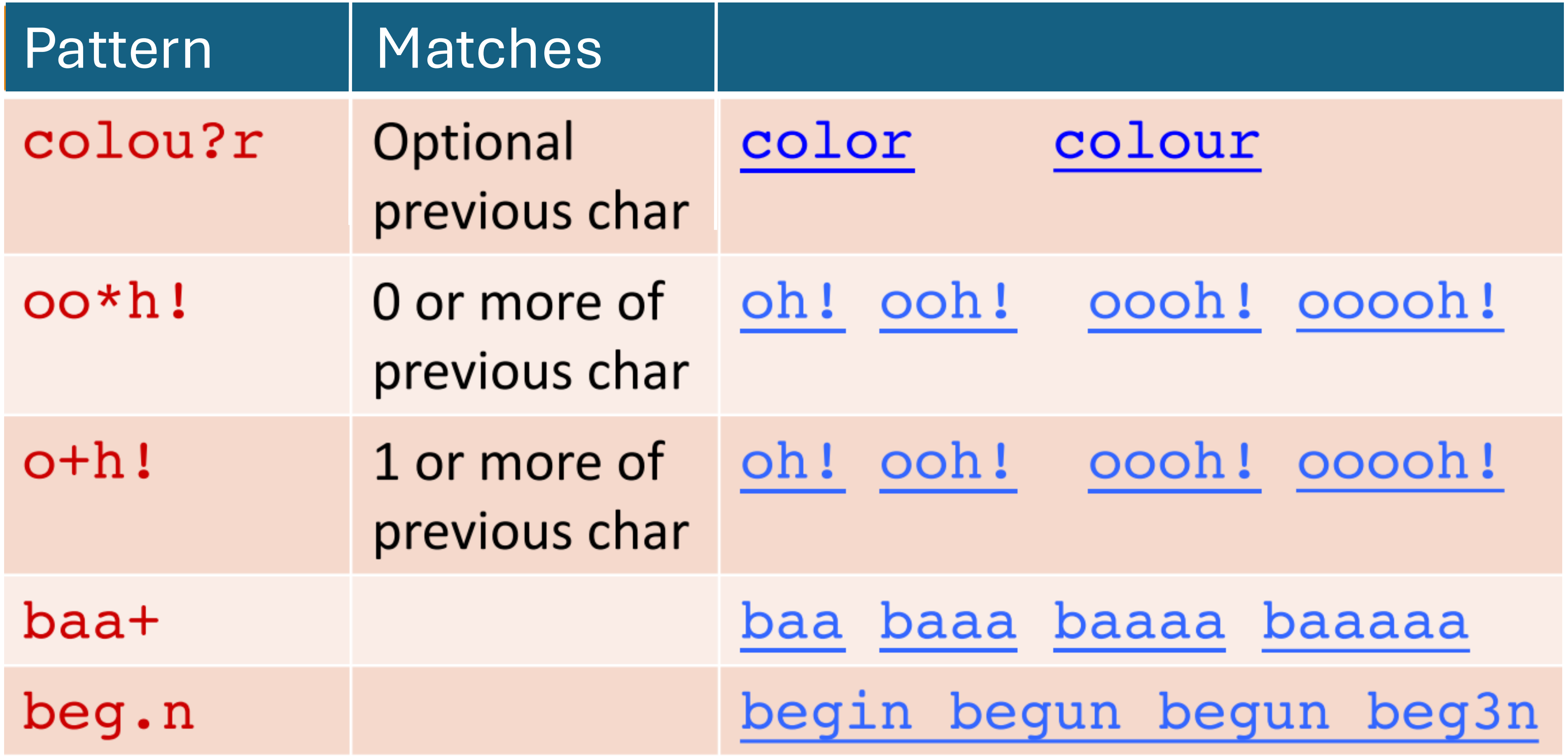
https://docs.python.org/3/library/re.html#module-re

Regular Expressions: More Disjunction (pipe |)

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

Slide Reference: https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc_2023.pdf

Regular Expressions: ? , Kleen operators(*+),.



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

Slide Reference: 2_TextProc_Mar_25_2021.pdf (stanford.edu)

Regular Expressions: Anchors ^ $

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

Slide Reference: https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc_2023.pdf

Some examples

Find me all instances of the word "the" in a text?

- the [Solution?]
  - Misses capitalized examples
- [tT]he [Solution?]
  - Incorrectly returns **other** or **Theology**

Slide Reference: https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc_2023.pdf

Some examples

Find me all instances of the word "the" in a text?

- the [Solution?]
  - Misses capitalized examples
- [tT]he [Solution?]
  - Incorrectly returns **other** or **Theology**

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

Slide Reference: https://web.stanford.edu/~jurafsky/slp3/slides/2_TextProc_2023.pdf

Activity-based on Regex

Given below the string, write a pattern to capture

Either e or ^

String Example : Look up ^ now

Activity-based on Regex

Given below the string, write a pattern to capture

Either e or ^

String Example : Look up ^ now

Solution: [e^]

Activity-based on Regex

Write a regular expression to match email addresses with the following criteria:

- The username part can contain letters (both uppercase and lowercase), numbers, dots (.), hyphens (-), and underscores (_).
- The domain part can contain letters (both uppercase and lowercase) and dots (.).
- The domain extension can only contain letters and must be between 2 to 4 characters long.

Activity-based on Regex

Write a regular expression to match email addresses with the following criteria:

- The username part can contain letters (both uppercase and lowercase), numbers, dots (.), hyphens (-), and underscores (_).
- The domain part can contain letters (both uppercase and lowercase) and dots (.).
- The domain extension can only contain letters and must be between 2 to 4 characters long.

Solution: `^[a-zA-Z0-9._-]+\@[a-zA-Z.]+\.[a-zA-Z]{2,4}$`

Activity-based on Regex

Solution: ^[a-zA-Z0-9._-]+\@[a-zA-Z.]+\.[a-zA-Z]{2,4}$

^ asserts the start of the string.

[a-zA-Z0-9._-]+ matches one or more alphanumeric characters, dots, hyphens, underscores, or hyphens before the "@" symbol.

@ matches the "@" symbol.

[a-zA-Z.]+ matches one or more alphabet characters, or dots in the domain name.

\. matches the dot separating the domain name and TLD.

[a-zA-Z]{2,4} matches the TLD consisting of at least 2 alphabetical characters.

$ asserts the end of the string.

Activity-based on Regex

Write a regular expression to match strings that represent valid time in 24-hour format (hh:mm:ss). Hours (hh) can range from 00 to 23, minutes (mm) and seconds (ss) can range from 00 to 59

Activity-based on Regex

Write a regular expression to match strings that represent valid time in 24-hour format (hh:mm:ss). Hours (hh) can range from 00 to 23, minutes (mm) and seconds (ss) can range from 00 to 59

Sol: ^(?:[01][0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9]$

Explanation:

^ asserts the start of the string.

(?:[01][0-9]|2[0-3]) matches hours in 24-hour format:

[01][0-9] matches hours from 00 to 19.

2[0-3] matches hours from 20 to 23.

: matches the colon separator between hours, minutes, and seconds.

[0-5][0-9] matches minutes and seconds from 00 to 59. The expression is repeated for minutes and seconds. $ asserts the end of the string.

Complete the Regex rules

Pattern	Expansion	Matches	Examples
\d		Any digit	
\D		Any non-digit	
\w		Any alphanumeric or _	
\W		Not alphanumeric or _	
\s		Whitespace (space, tab)	
\S		Not whitespace	

Complete the Regex rules

Pattern	Expansion	Matches	Examples
\d	[0-9]	Any digit	Fahreneit 451
\D	[^0-9]	Any non-digit	Blue Moon
\w	[a-ZA-Z0-9_]	Any alphanumeric or _	Daiyu
\W	[^\w]	Not alphanumeric or _	Look!
\s	[\r\t\n\f]	Whitespace (space, tab)	Look␣up
\S	[^\s]	Not whitespace	Look up

\f refers to feed character in the string. For example: page breaks

A note about Python regular expressions

◦ Regex and Python both use backslash "\" for special characters. You must type extra backslashes!

◦ "\\d+" to search for 1 or more digits

◦ "\n" in Python means the "newline" character, not a "slash" followed by an "n". Need "\\n" for two characters.

◦ Instead: use Python's raw string notation for regex:

◦ r"[tT]he"

◦ r"\d+" matches one or more digits

◦ instead of "\\d+"

False positives and false negatives

- The process we just went through was based on fixing two kinds of errors:

1. Not matching things that we should have matched (The) **False negatives**

2. Matching strings that we should not have matched (there, then, other) **False positives**

Characterizing work on NLP

In NLP we are always dealing with these kinds of errors. Reducing the error rate for an application often involves two antagonistic efforts:

- **Increasing coverage (or recall)** (minimizing false negatives).
- **Increasing accuracy (or precision)** (minimizing false positives)

Regular expressions play a surprisingly large role

Widely used in both academics and industry:

1. Part of most text processing tasks, even for big neural language model pipelines
    - including text formatting and pre-processing
2. Very useful for data analysis of any text data

Class Activity

Goal: To match regular expressions

Task: To generate a large dataset of student records and write a Python program to match specific regular expressions against the data.

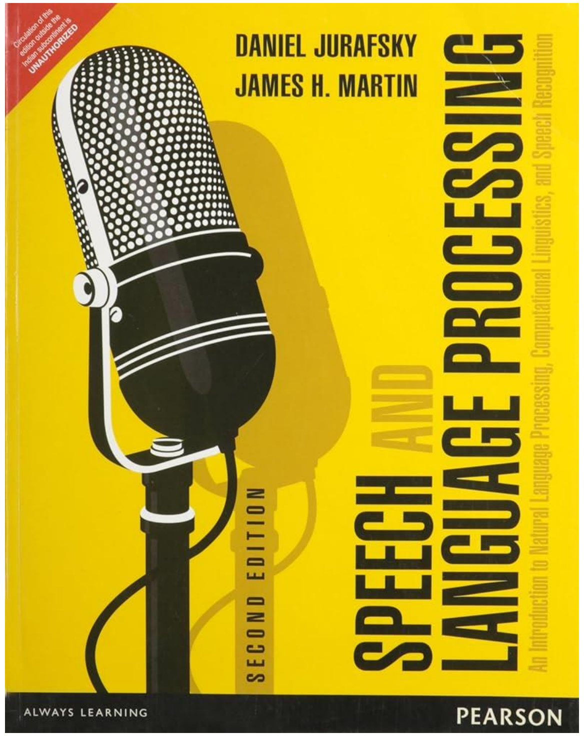
You will generate a dataset consisting of 100,000 student records. Each record will contain the following fields:

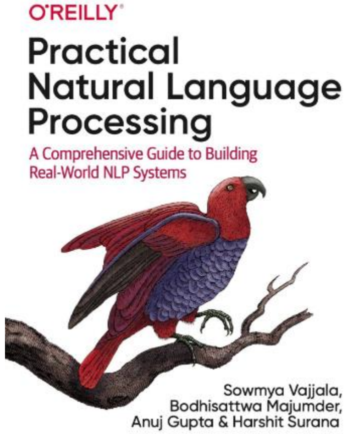
- **Student Name:** A random name composed of a first and last name.
- **Roll Number:** A unique identifier, such as SE22UARI001 (where "SE22UARI" is a batch code and the digits are a sequence).
- **Courses Taken:** A list of 3-5 courses represented by course codes like CS3126, CS3202, etc.
- **Email:** A randomly generated email address associated with the student with domain name @mahindrauniversity.edu.in.
- **Section:** The section AI1,AI2, AI3 etc.

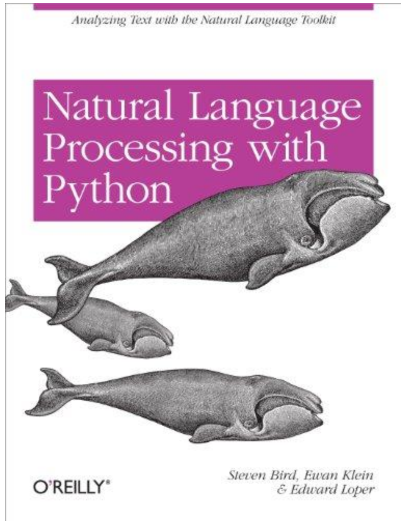
Outcome: By the end of this activity, you should be able to: 1. Understand the use of regular expressions in data filtering. 2. Generate large datasets programmatically. 3. Apply regular expressions effectively to extract meaningful information from datasets.

Reference materials

- https://vlanc-lab.github.io/mu-nlp-course/teachings/fall-2024-AI-nlp.html
- Lecture notes
- (A) Speech and Language Processing by Daniel Jurafsky and James H. Martin
- (B) Natural Language Processing with Python. (updated edition based on Python 3 and NLTK
- 3) Steven Bird et al. O’Reilly Media


Circulation of this edition outside the Indian subcontinent is UNAUTHORIZED
DANIEL JURAFSKY
JAMES H. MARTIN
SECOND EDITION
SPEECH AND LANGUAGE PROCESSING
An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition
ALWAYS LEARNING
PEARSON


O'REILLY
Practical Natural Language Processing
A Comprehensive Guide to Building Real-World NLP Systems
Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta & Harshit Surana


Analyzing Text with the Natural Language Toolkit
Natural Language Processing with Python
O'REILLY
Steven Bird, Ewan Klein & Edward Loper