ALGONQUIN COLLEGE

CST8507: NATURAL LANGUAGE PROCESSING

WEEK#2
TEXT PREPROCESSING AND EXPLORATORY ANALYSIS

DEVELOPED BY
HALA OWN, PH.D.

Lesson Agenda

- ❑Regular expression
- ❑Tokenization
- ❑Stemming
- ❑Noise Entities Removal
- ❑Part Of Speech (POS) tagging
- ❑Named Entity Recognition

Approaches to NLP

- **Heuristics-Based NLP**
  - Regular Expression
- **Machine Learning for NLP**
  - Supervised
  - Unsupervised
- **Deep Learning for NLP**
  - Recurrent neural networks
  - Long short-term memory
- **Transformers**

Rule Based System

Computer Troubleshooting

Rule 1: If the computer does not power on, check if the power cable is connected.

Rule 2: If the power is on but the screen is blank, check the monitor's connections.

Rule 3: If there is no sound, check the speaker connections and volume settings.

Rule 4: If the computer is slow, check for malware and free up disk space.

- Design a simple rule-based inference engine to match user-reported symptoms with corresponding rules and provide recommendations.

User Query Example

```
User: My computer is not powering on.
System: Recommendation - Check if the power cable is connected.
```

What are Regular Expressions?

- ❑ In computing, a regular expression, also referred to as **"regex"** or **"regexp",** provides a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters.
- ❑ A regular expression is written in a formal language that can be interpreted by a regular expression processor.

Regular Expression Quick Guide:metacharacters

- ❑ . Matches any single character
- ❑ [ ] Matches a single character in the listed set
- ❑ ^ Beginning of string(based on the position)
- ❑ $ End of string
- ❑ * matches 0 or more characters
- ❑+ matches 1 or more characters
- ❑ ? zero or one occurrence of the preceding character

Regular Expression Quick Guide…

- ❑ **{ m,n}** specify number of times character is matched between m and n times
- ❑ \ escape character
- ❑ **|** or
- ❑ **( )** capture group inside parenthesis

Character Classes

- **\s** - matches any whitespace
- **\w** - matches any alpha character. Equivalent to [A-Za-z]
- **\d** - matches any numeric character. Equivalent to [0-9]

- **You may negate these by capitalizing. For example, \D matches anything not a digit**

Regular Expressions: Examples

Letters inside square brackets []

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

Ranges [A-Z]

Pattern	Matches
[A-Z]	An upper case letter
[a-z]	A lower case letter
[0-9]	A single digit

Source: Text book

Regular Expressions: ? *+.

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

Source: Text book

Regular Expressions: Negation

Negations `[^Ss]`

Carat means negation only when first in []

Pattern	Matches
`[^A-Z]`	Not an upper case letter
`[^Ss]`	Neither ‘S’ nor ‘s’
`[^e^]`	Neither e nor ^

Source: Text book

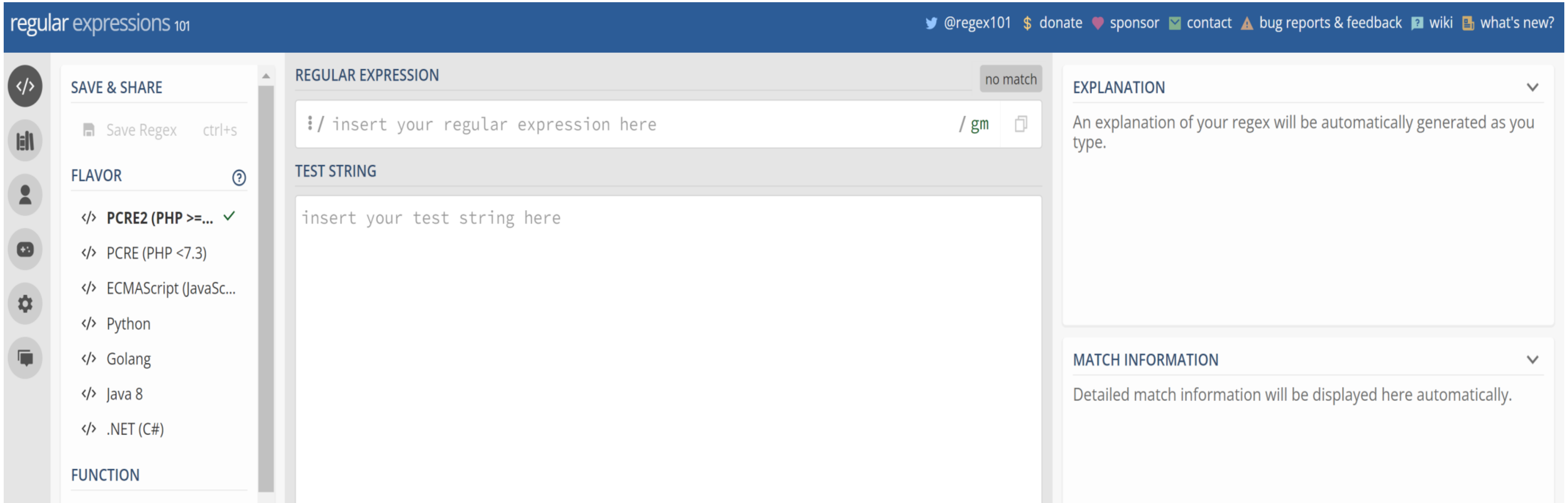
Regular Expressions: Anchors ^ $

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

Source: Text book

Online Regular Expressions

- https://regex101.com/



An explanation of your regex will be automatically generated as you type.

Detailed match information will be displayed here automatically.

Python Regex Functions

- ❑ `re.match(r, s)` returns a matched object if the regex **r** matches **at the start** of string **s**

- ❑ `re.search(r, s)` returns a matched object if the regex **r** matches **anywhere in string s**

- ❑ `findall(pattern, string )` return a list of strings giving all nonoverlapping matches of pattern in string.

Python Regex Functions…

- **`sub(pattern, repl, string)`** returns the string obtained by replacing the (first count) leftmost nonoverlapping occurrences of pattern (a string or a pattern object) in string by **repl.**

- **`compile(pattern )`** compiles a regular expression pattern string into a regular expression **pattern object**, for later matching.

Python Regex Functions…

- ❑ **groups()** Returns a tuple of all group’s substrings of the match .

- ❑ **span([group])** Returns the two-item tuple: (start(group),end(group))

Python Regex Functions…

```
import re
re.split(" ", "ab bc cd")
['ab', 'bc', 'cd']
re.split("\d", "ab1bc4cd")  ['ab', 'bc', 'cd']
```

Regular Expression: Use Cases

- Text cleaning
- Tokenization
- Information Retrieval
- Sentiment Analysis
- Language Detection

Class Activity(work on groups)

Q:Write a regexp to check if any URL exists in the text. Test your solution with the following text

```
text = "Visit my website at https://www.example.com or check out http://another-example.org/path/page.html"
```

Class Activity(work on groups)…

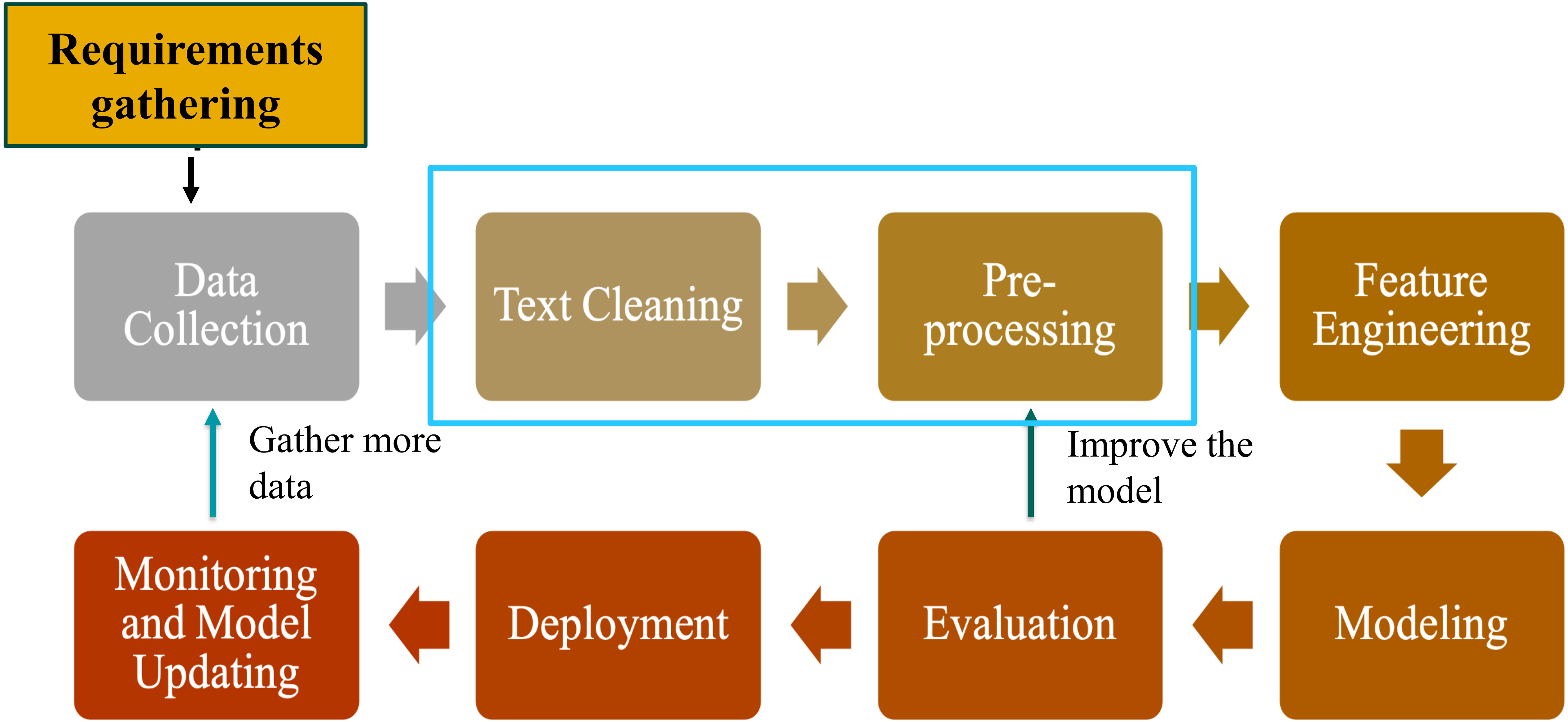
- Given a text, list all the longest possible substrings that are proper variable names in most of the programming languages. A proper variable name is defined as the one that does not start with a digit and does not contain any special character other than under score, and it can have arbitrary number of characters.
- Test your solution with the following text

```
Text='hsdgkjdh;efjewipjrndendrwerji2;;;;8888p9nskdj3905jdkwqld***w3w94
5{{{{{jwkqs ;weoijrtwioejri’
```

The output

```
['hsdgkjdh', 'efjewipjrndendrwerji2', 'p9nskdj3905jdkwqld', 'w3w945', 'jwkqs',
'weoijrtwioejri']
```

NLP Development Life Cycle



Requirements gathering

Data Collection

Text Cleaning

Pre-processing

Feature Engineering

Modeling

Evaluation

Deployment

Monitoring and Model Updating

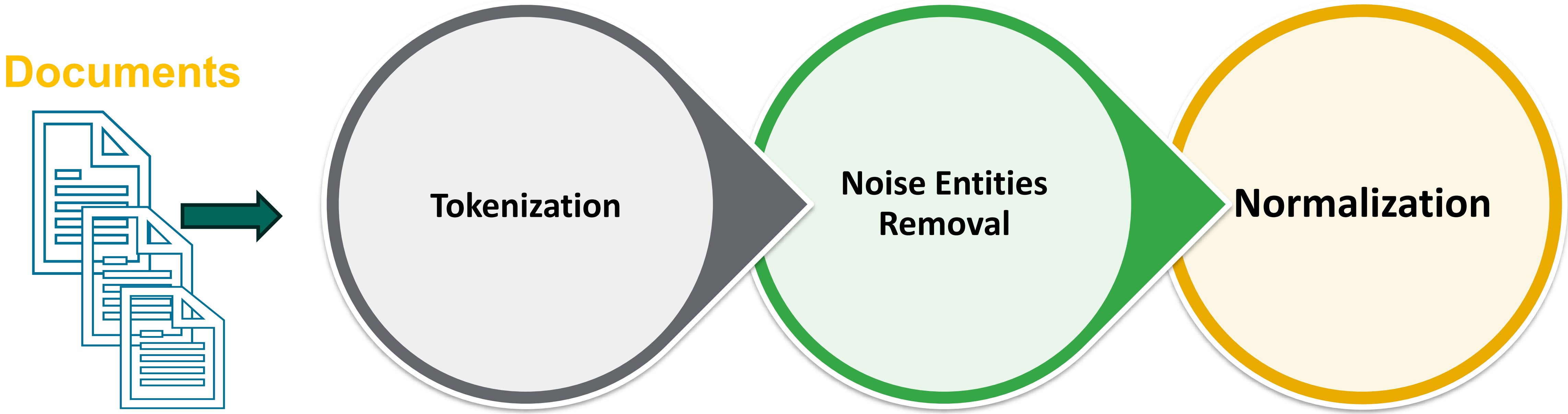
Gather more data

Improve the model

Text-Preprocessing and Cleaning :Motivations

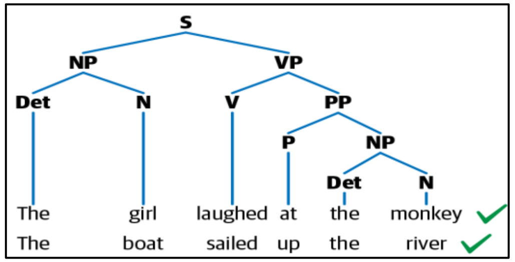
- **Clean** And standardize the text data to make it more suitable for NLP tasks.
- **Convert** The text data into A format that can be easily understood and processed by NLP algorithms.
- **Improve** The performance and accuracy of NLP models.

Text Pre-Processing Pipeline



May be varied depending on the task you are working on and the data you have

Building Blocks of Language

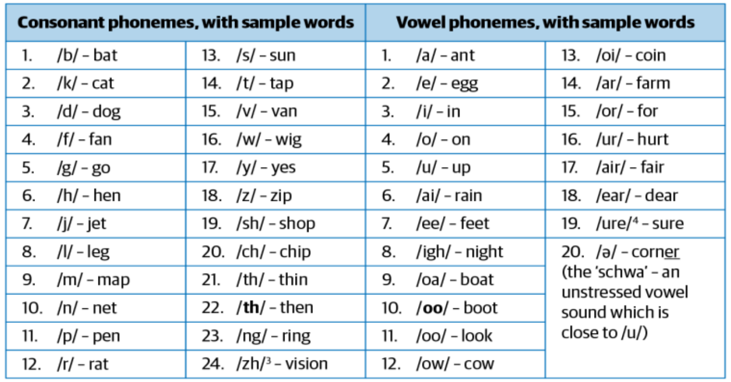

S
NP
VP
Det
N
V
PP
P
NP
Det
N
The girl laughed at the monkey ✓
The boat sailed up the river ✓

unbreakable
un + break + able

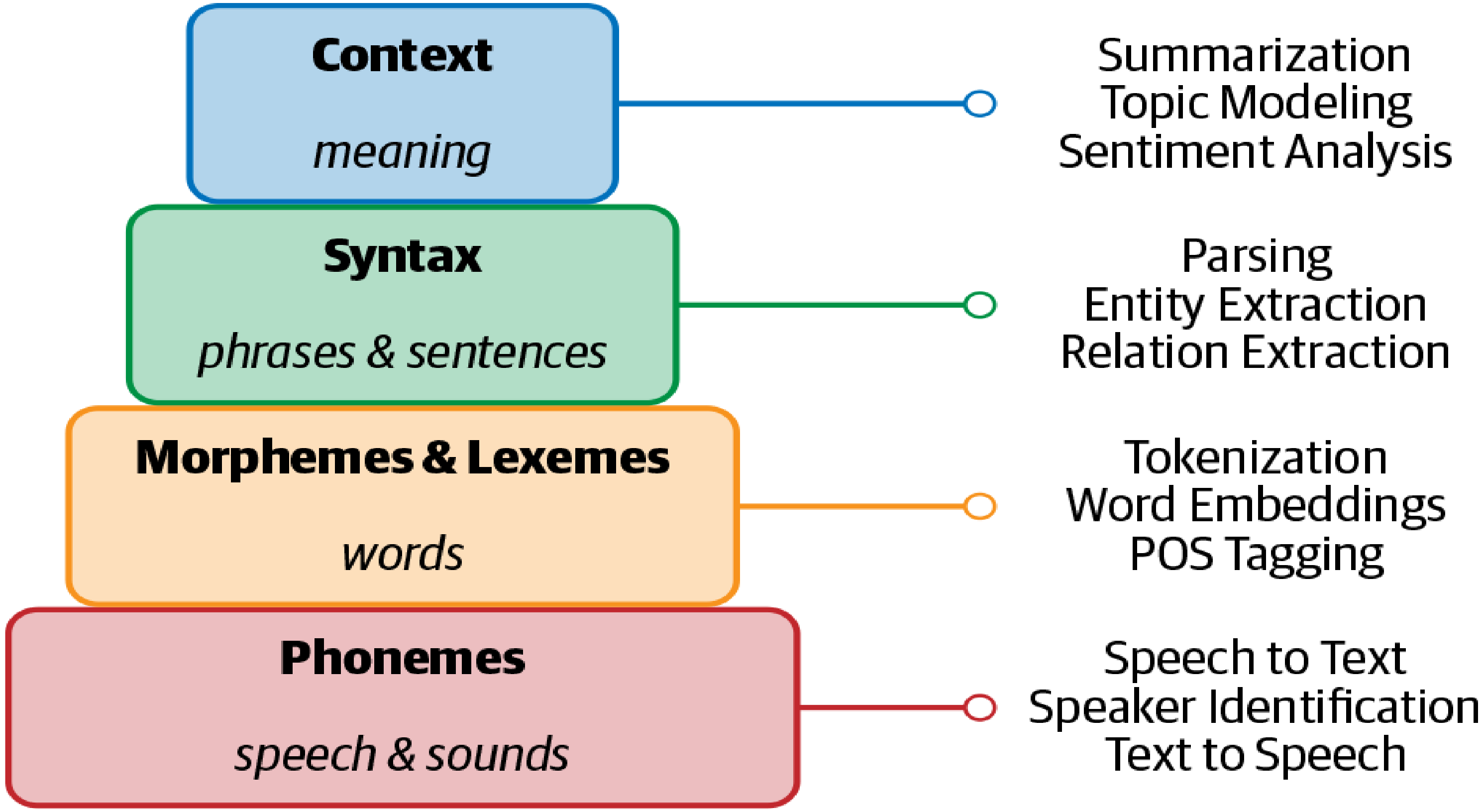
cats
cat + s

tumbling
tumble + ing

unreliability
un + rely + able + ity



Consonant phonemes, with sample words		Vowel phonemes, with sample words	
1. /b/ – bat	13. /s/ – sun	1. /a/ – ant	13. /oi/ – coin
2. /k/ – cat	14. /t/ – tap	2. /e/ – egg	14. /ar/ – farm
3. /d/ – dog	15. /v/ – van	3. /i/ – in	15. /or/ – for
4. /f/ – fan	16. /w/ – wig	4. /o/ – on	16. /ur/ – hurt
5. /g/ – go	17. /y/ – yes	5. /u/ – up	17. /air/ – fair
6. /h/ – hen	18. /z/ – zip	6. /ai/ – rain	18. /ear/ – dear
7. /j/ – jet	19. /sh/ – shop	7. /ee/ – feet	19. /ure/^{4} – sure
8. /l/ – leg	20. /ch/ – chip	8. /igh/ – night	20. /ə/ – corner (the 'schwa' – an unstressed vowel sound which is close to /u/)
9. /m/ – map	21. /th/ – thin	9. /oa/ – boat	
10. /n/ – net	22. /**th**/ – then	10. /**oo**/ – boot	
11. /p/ – pen	23. /ng/ – ring	11. /oo/ – look	
12. /r/ – rat	24. /zh/^{3} – vision	12. /ow/ – cow	



Blocks of Language

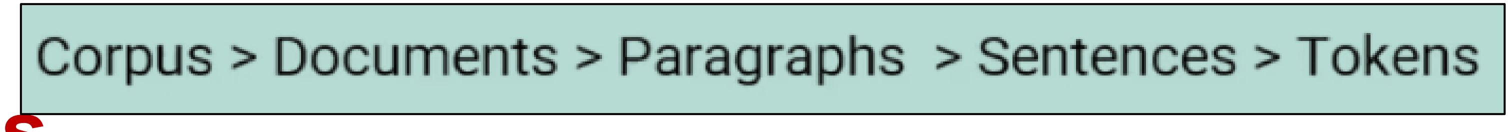
Applications

Text Preprocessing: Basic Terminology

❑Corpus

A Corpus is defined as a collection of text documents.

- A data set containing news.
- The tweets containing Twitter.

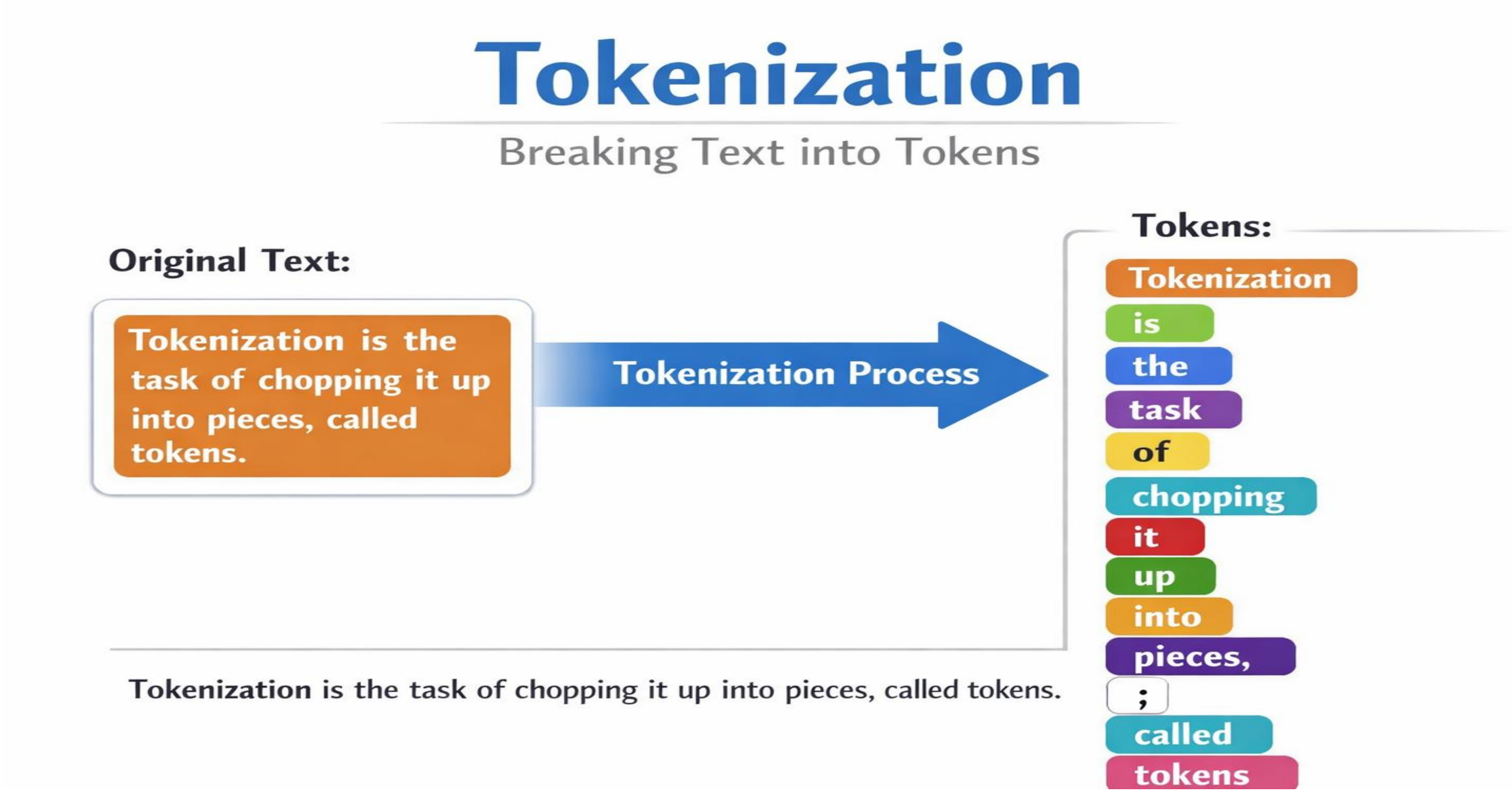


Corpus > Documents > Paragraphs > Sentences > Tokens

❑Words

- unit of language that has a specific **meaning** and is separated by spaces or punctuation.

Tokenization



Tokenization

Breaking Text into Tokens

Original Text:

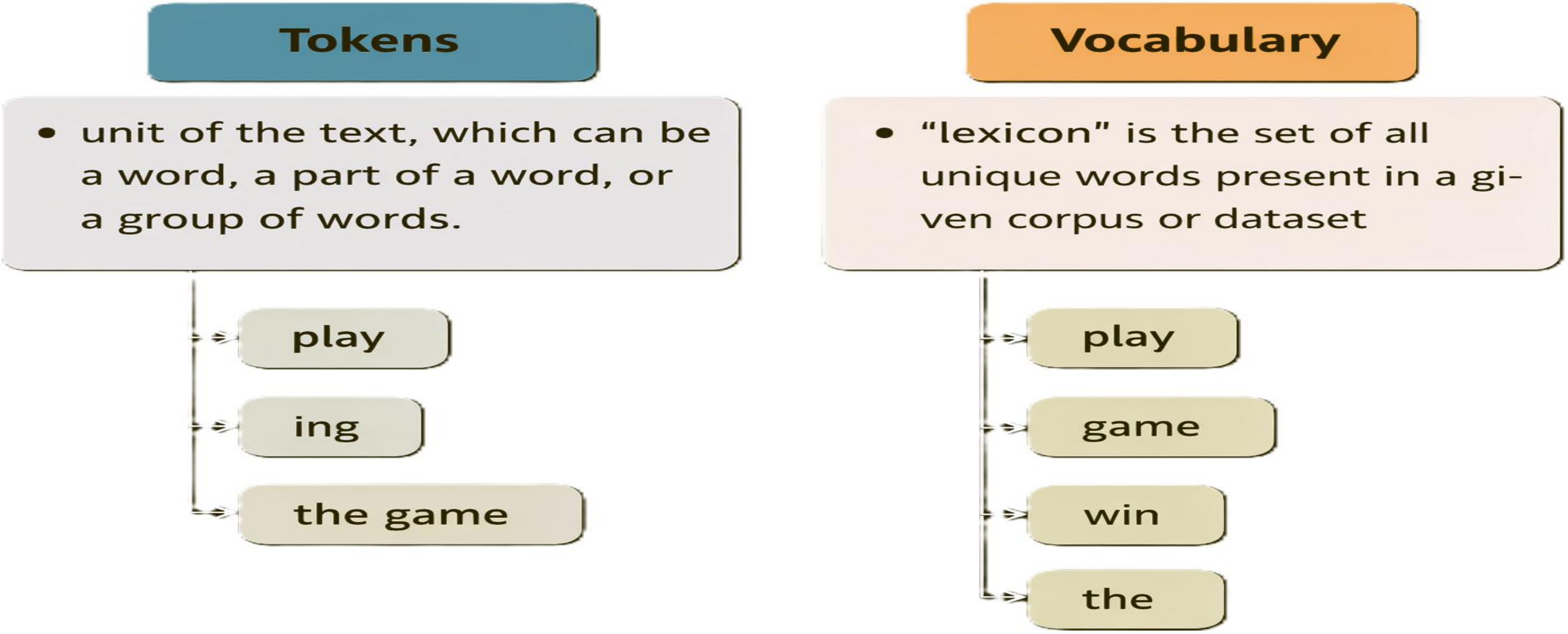
Tokenization is the task of chopping it up into pieces, called tokens.

Tokenization Process

Tokenization is the task of chopping it up into pieces, called tokens.

Image created by ChatGPT

Text Pre-processing: Basic Terminology…



Tokens

- unit of the text, which can be a word, a part of a word, or a group of words.

play

ing

the game

Vocabulary

- “lexicon” is the set of all unique words present in a given corpus or dataset

play

game

win

the

Image created by ChatGPT

Tokenization

- **Demo**

https://text-processing.com/demo/tokenize/

Noise Entities Removal(Cleaning Data)

Noise is considered as that piece of text which is **not relevant** to the context of the data .

- **Removing Capital letters**

```
lowercased_text = text.lower()
```

- **Removing Numbers**

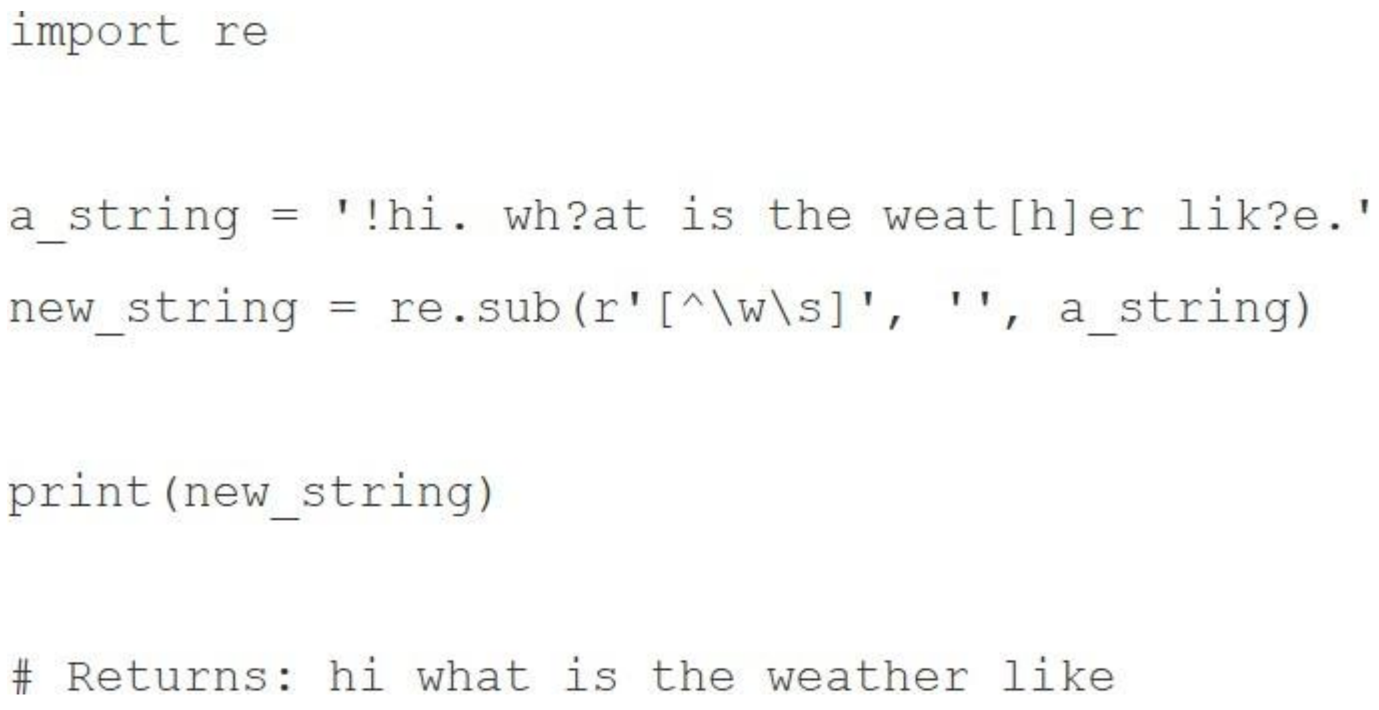
```
clean_text = re.sub('\w*\d\w*', ' ', clean_text)
```

- **Removing Punctuation**
- **Removing stop words**

Cleaning Data…

Demo

Cleaning Data - Punctuations



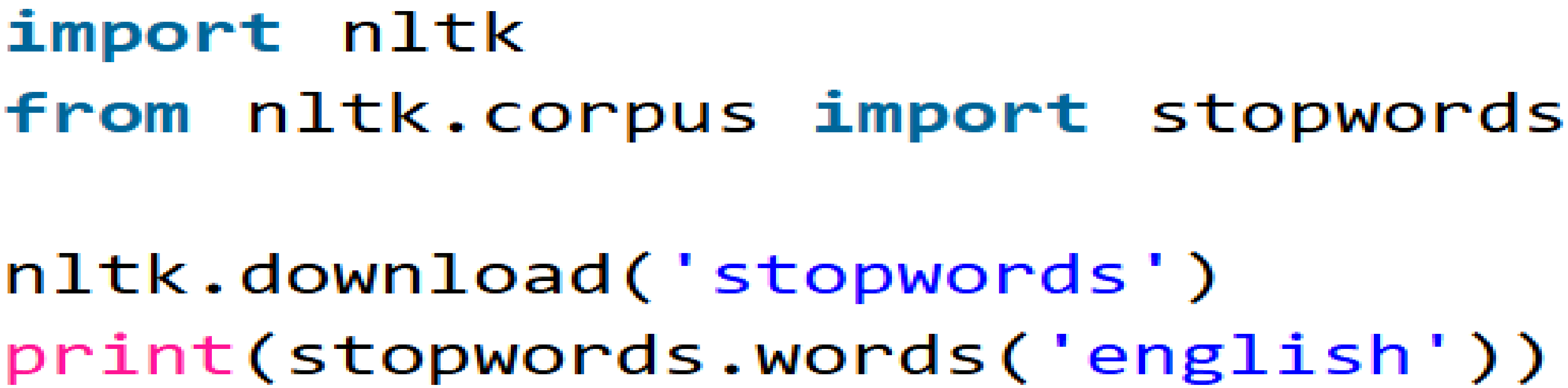
```
import re


a_string = '!hi. wh?at is the weat[h]er lik?e.'
new_string = re.sub(r'[^\w\s]', '', a_string)


print(new_string)


# Returns: hi what is the weather like
```

Cleaning Data – Stop words



```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
print(stopwords.words('english'))
```

Cleaning Data…

Language stop words:

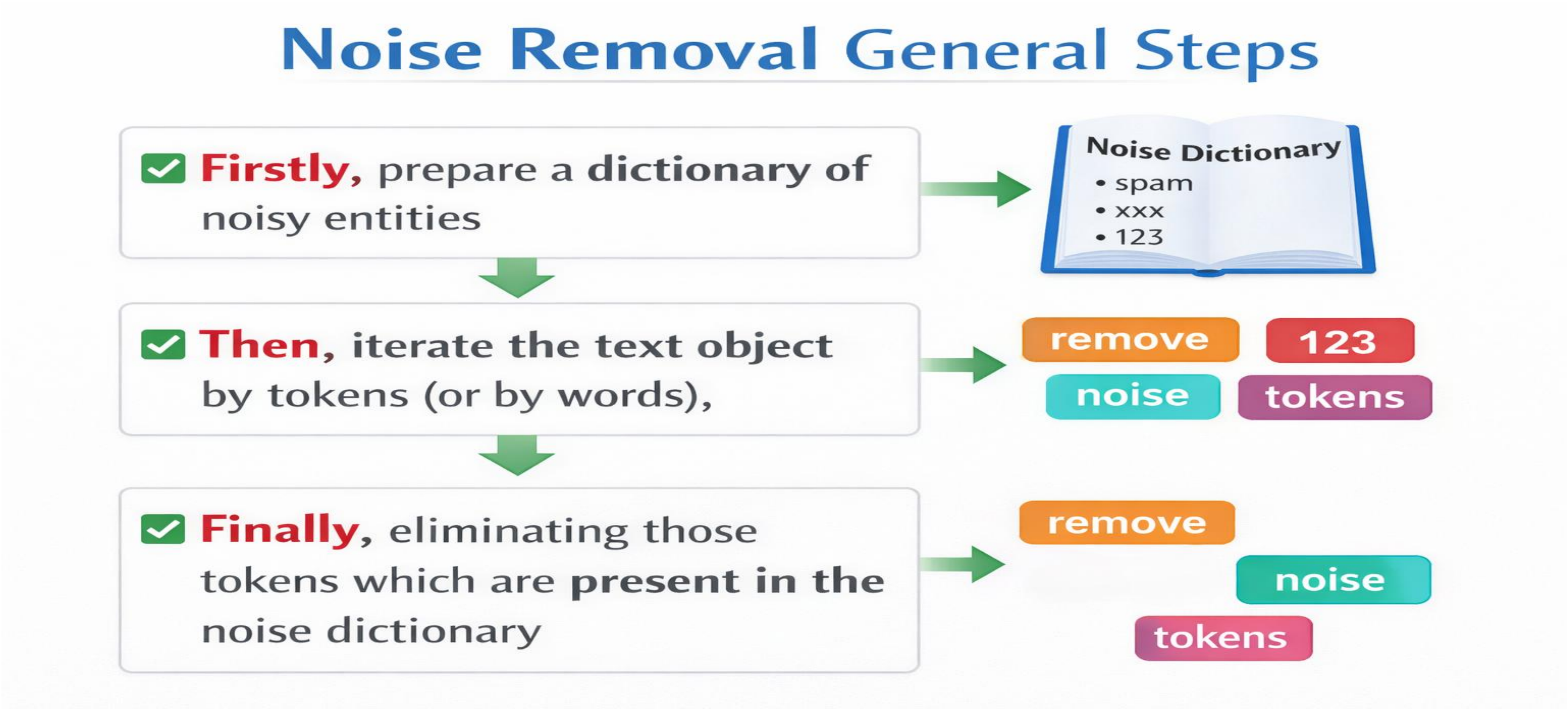
Demo

Other Noise Entities

WWW
URLs or links
@#
Social media entities
happy face
Replacing emoticons with words
compound Term
Compound Term

Image created by ChatGPT

Noise Removal General Steps



Noise Removal General Steps

- ☑ **Firstly,** prepare a dictionary of noisy entities
- ☑ **Then,** iterate the text object by tokens (or by words),
- ☑ **Finally,** eliminating those tokens which are present in the noise dictionary

Image created by flowcastGPT with some added updates

Compound Term Extraction

- Extracting and tagging compound words or phrases in text

- **Demo**

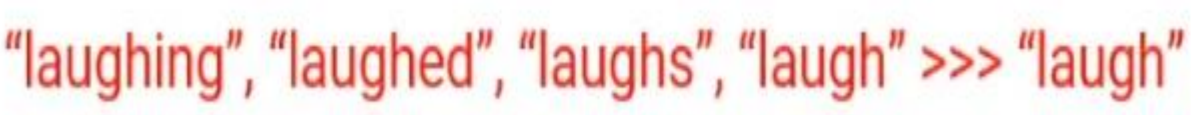
What is Normalization?

- Normalization is the process of converting a token into its base form.
- Inflection from a word is removed

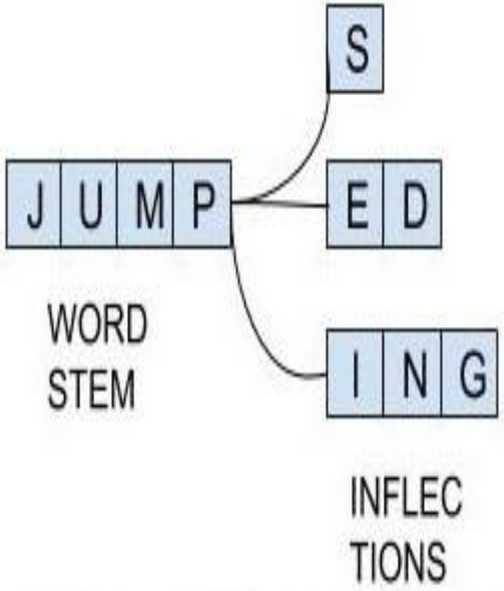
Stemming

- **Word stems**, known as the base form of a word.

Example:



“laughing”, “laughed”, “laughs”, “laugh” >>> “laugh”



Word stem and its inflections (Source: Text Analytics with Python, Apress/Springer 2016)

Stemming Algorithms(NLTK)

- **Porter Stemmer**
- Snowball Stemmer
- Lancaster Stemmer

Stemming: Applications

- Classifying text
- Clustering text, and
- Information retrieval, etc.

Lemmatization

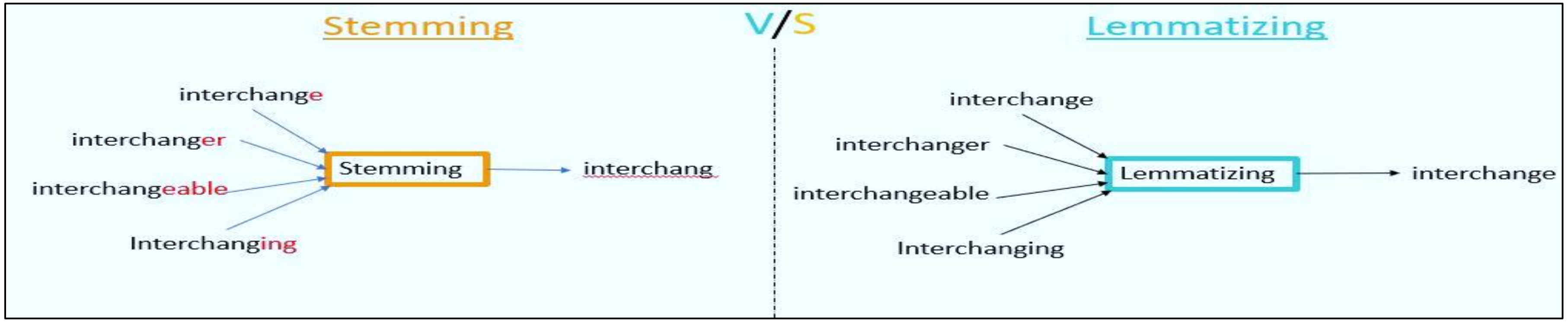
Obtaining the root form of the word, as it makes use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations).

The output of lemmatization is the root word called lemma

Example:

```
Am, Are, Is >> Be
Running, Ran, Run >> Run
```

Normalization Techniques



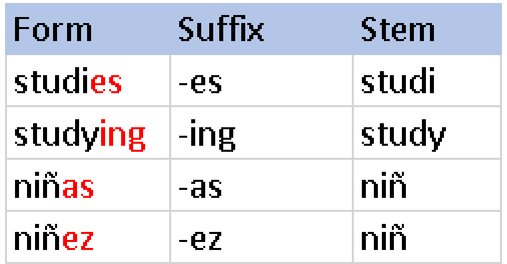
- ❑ Lemmatization is a potentially more accurate way to normalize a word than stemming, because it takes into account a **word’s meaning.**
- ❑ A lemmatizer uses a knowledge base of word synonyms and word endings to ensure that only words that mean similar things are consolidated into a single token.

https://www.pluralsight.com/guides/importance-of-text-pre-processing

Example of Difference between Stemming and Lemmatization

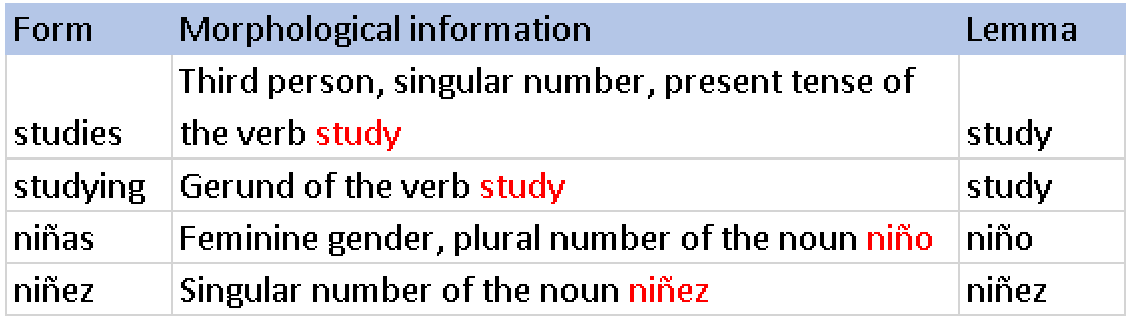
- Based on Context Consideration
  - Stemming is Typically faster but not that accurate
  - Lemmatization is typically more Accurate
- Speed vs Accuracy trade-off

stemming



Form	Suffix	Stem
studies	-es	studi
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

lemmatization



Form	Morphological information	Lemma
studies	Third person, singular number, present tense of the verb study	study
studying	Gerund of the verb study	study
niñas	Feminine gender, plural number of the noun niño	niño
niñez	Singular number of the noun niñez	niñez

Lemmatization Tools

- Wordnet Lemmatizer(NLTK)
- Spacy Lemmatizer
- TextBlob
- CLiPS Pattern
- Stanford CoreNLP
- Gensim Lemmatizer
- TreeTagger

When Not to Use Lemmatization and Stemming

- Specific tasks
- Computational cost
- Social media

Importance of Normalization

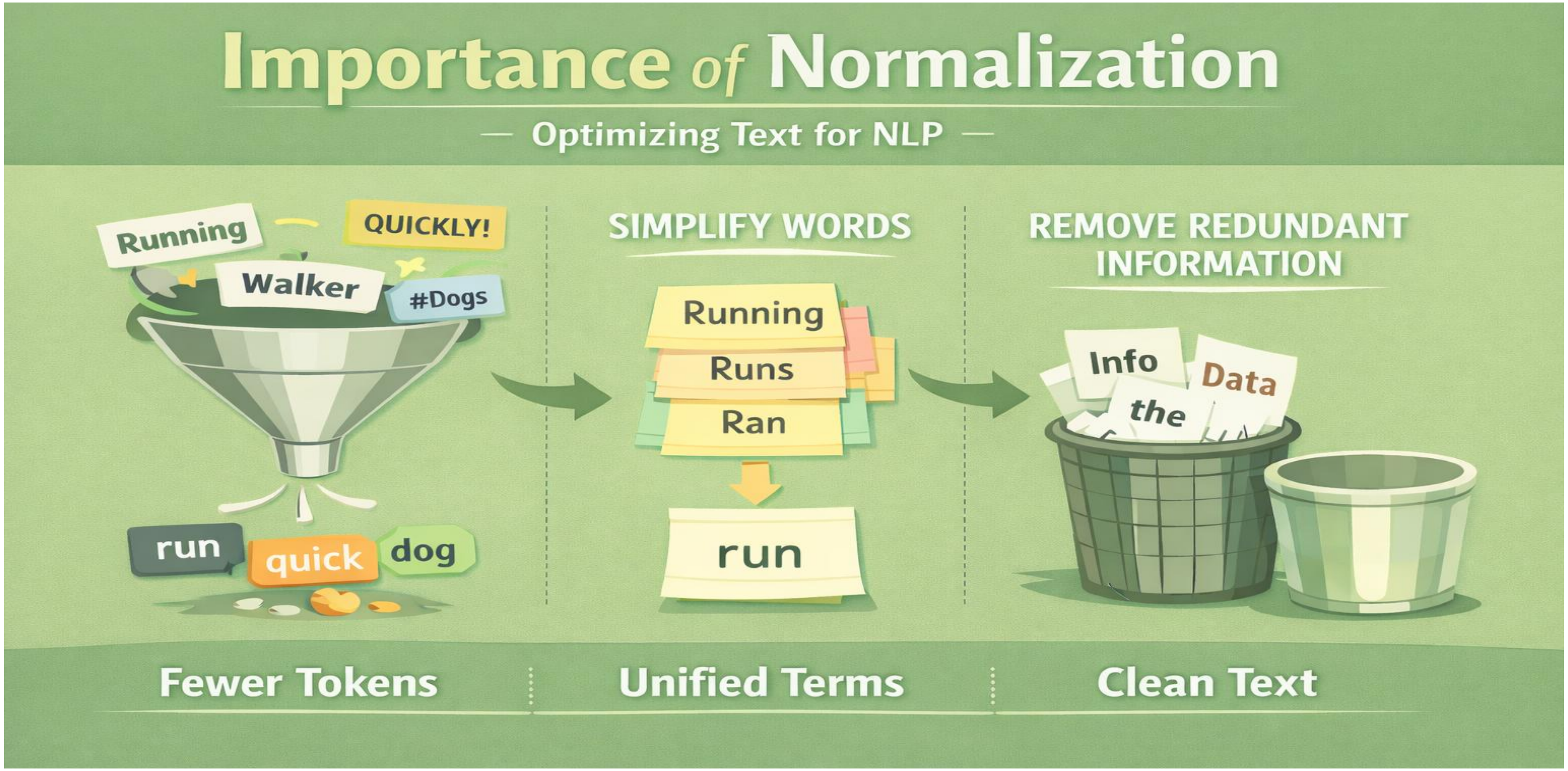


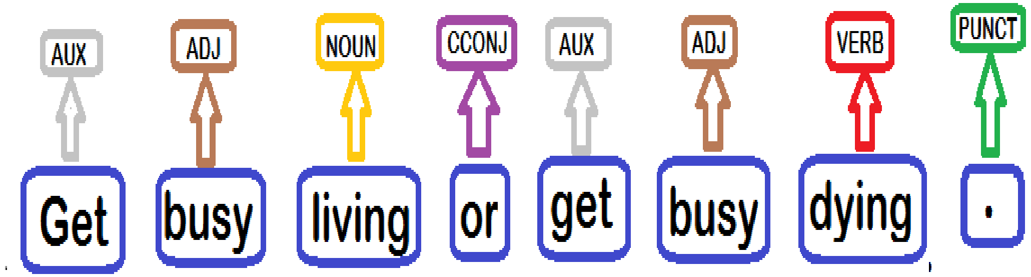
Image created by ChatGPT

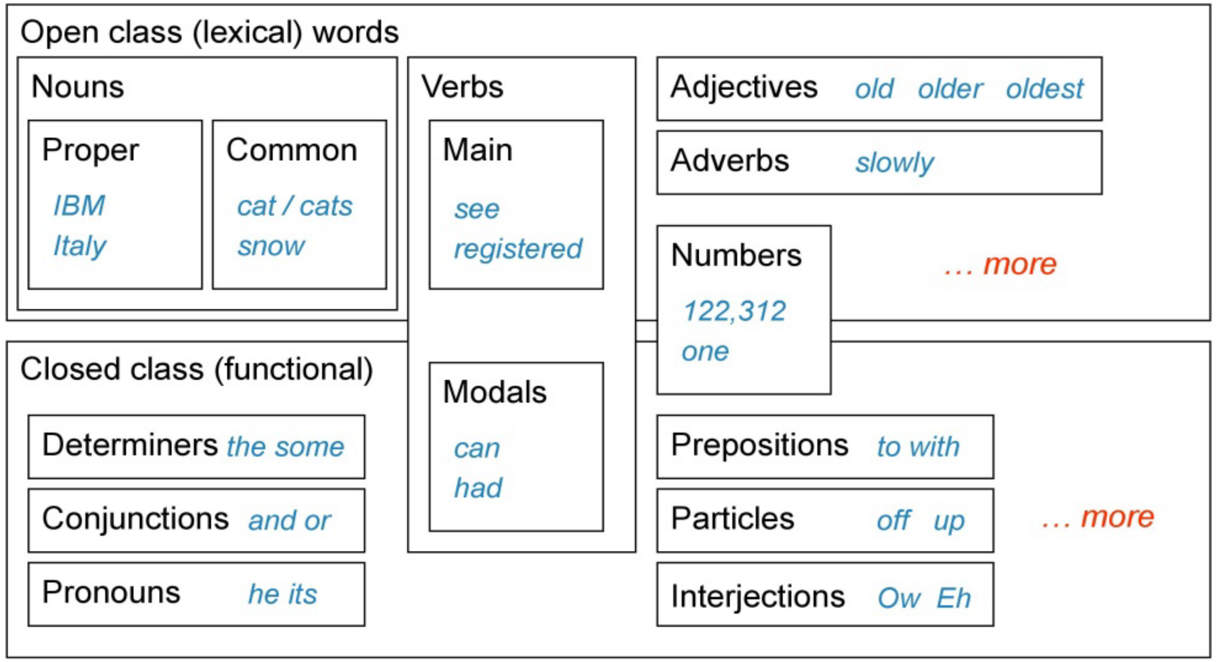
How Do They Work?

- ❑ Demo

Parts of Speech (POS) Tagging

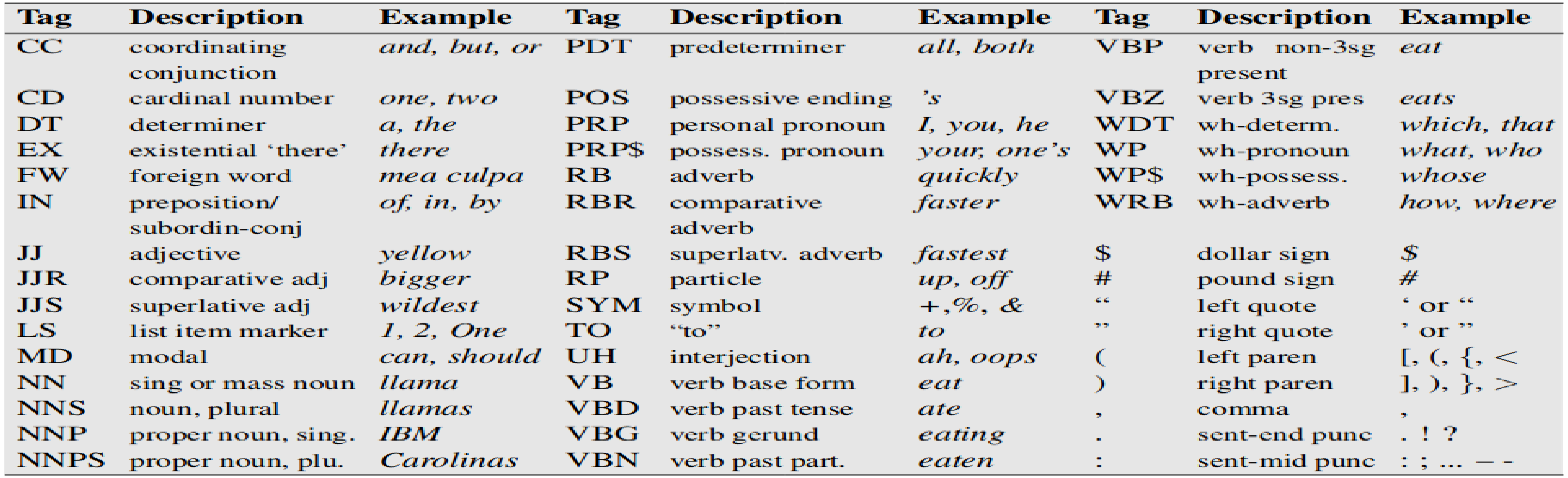
- process of identifying a word as nouns, pronouns, verbs, adjectives, etc.





https://nlpforhackers.io/tag/part-of-speech/

Parts of Speech (POS) Tagging



Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*’s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential ‘there’	*there*	PRP$	possess. pronoun	*your, one’s*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... — -

You can print it Using Python

```
>>>nltk.help.upenn_tagset()
```

https://thottingal.in/blog/2019/09/10/bis-pos-tagset-review/

Parts of Speech (POS) Tagging

Part-of-Speech Tagging | Demo

Why Do We Need Part Of Speech (POS)?

- ❑Syntactic and semantic analysis.
- ❑Structure and meaning of sentences.
    - **improve the accuracy of other NLP tasks**

Named Entity Recognition

- Identifies and tags named entities in text (people, places, organizations, phone numbers, emails, etc.)

```
from nltk.chunk import ne_chunk
text="James Smith lives in the United States."
tokens = pos_tag(word_tokenize(text))
entities = ne_chunk(tokens)
```

Why Do We Need Named Entity Recognition

- Information extraction
- Searching and indexing
- Sentiment analysis

Class Activity

For which of the following tasks we shouldn't do stemming/lemmatization?

A. Poetry Analysis
B. Text Classification
C. Sentiment Analysis

Join at

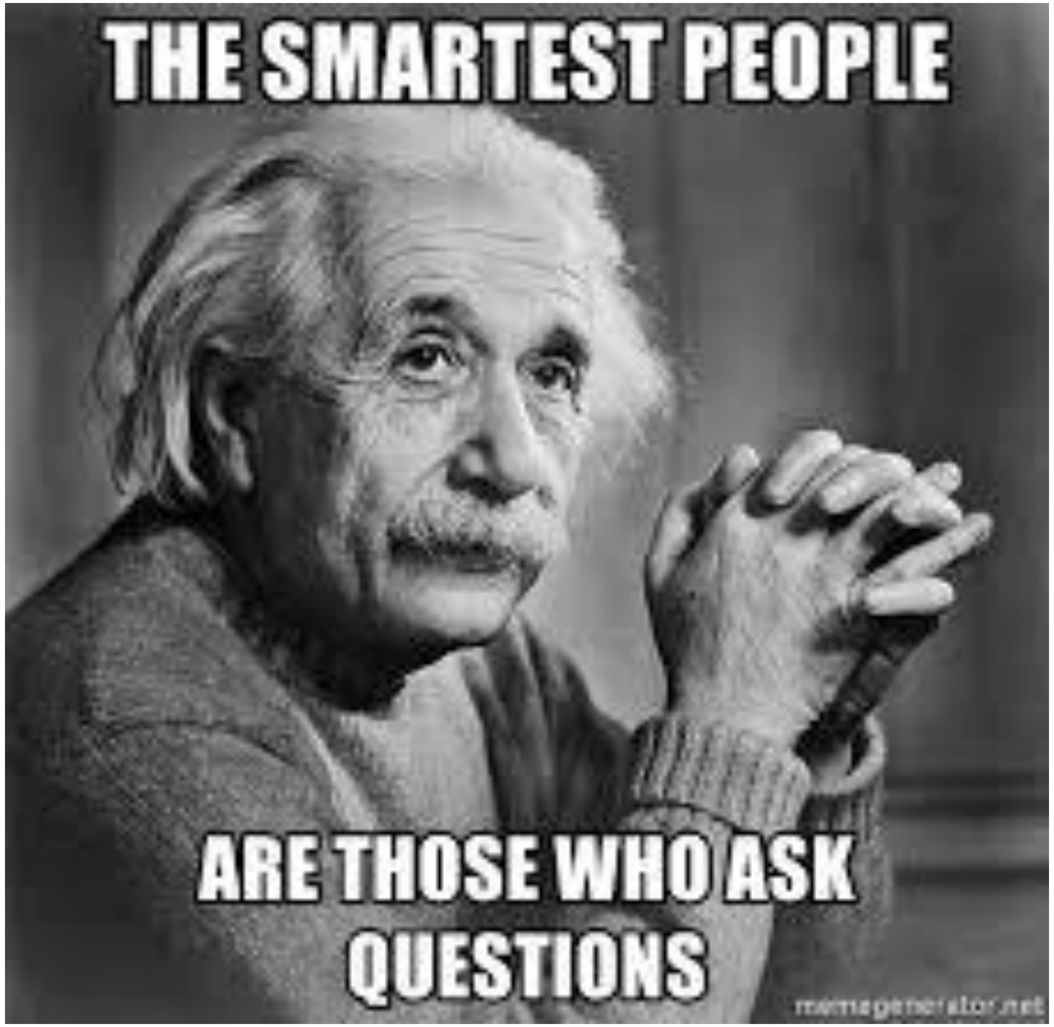
slido.com

#3175 033

Summary

- **Regular expressions**, which will play an important part throughout the course
- **Fundamental operations in text analysis:**
  - tokenization: breaking up a character string into words, punctuation marks and other meaningful expressions;
  - stemming: removing affixes from words
  - tagging: associating each word in a text with a grammatical category or part of speech.

Q&A


THE SMARTEST PEOPLE
ARE THOSE WHO ASK QUESTIONS
memegenerator.net