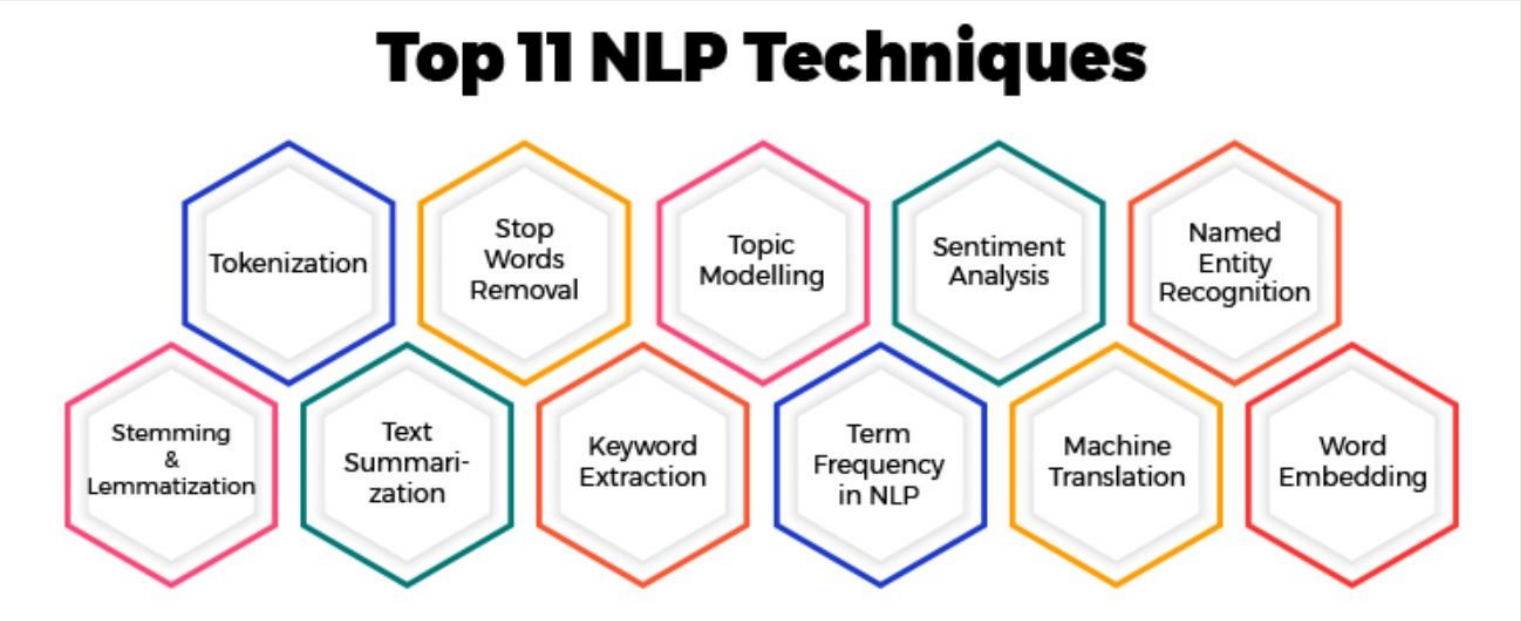
Natural Language Pre-processing



Prepared by: Syed Afroz Ali

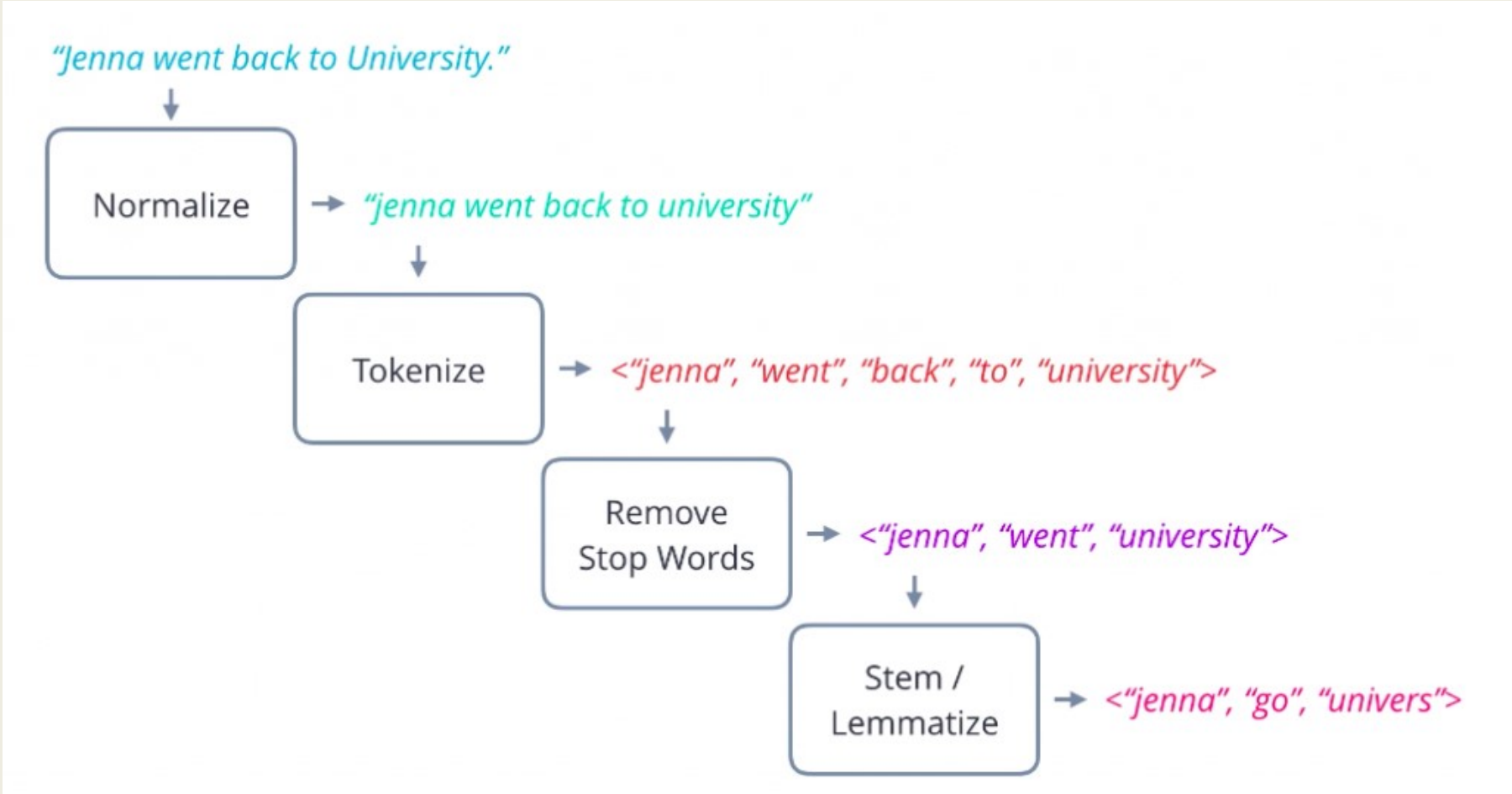
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832

Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

Natural Language Pre-processing (NLP)

Natural Language Processing (NLP) involves a series of pre-processing steps to transform raw text data into a format suitable for analysis or machine learning models. These steps help improve the quality of the data and make it easier for algorithms to understand and process the text. Below are the key pre-processing steps used in NLP.

Common Pre-processing step commonly used before feeding data into an NLP model


"Jenna went back to University."
Normalize
"jenna went back to university"
Tokenize
<"jenna", "went", "back", "to", "university">
Remove Stop Words
<"jenna", "went", "university">
Stem / Lemmatize
<"jenna", "go", "univers">

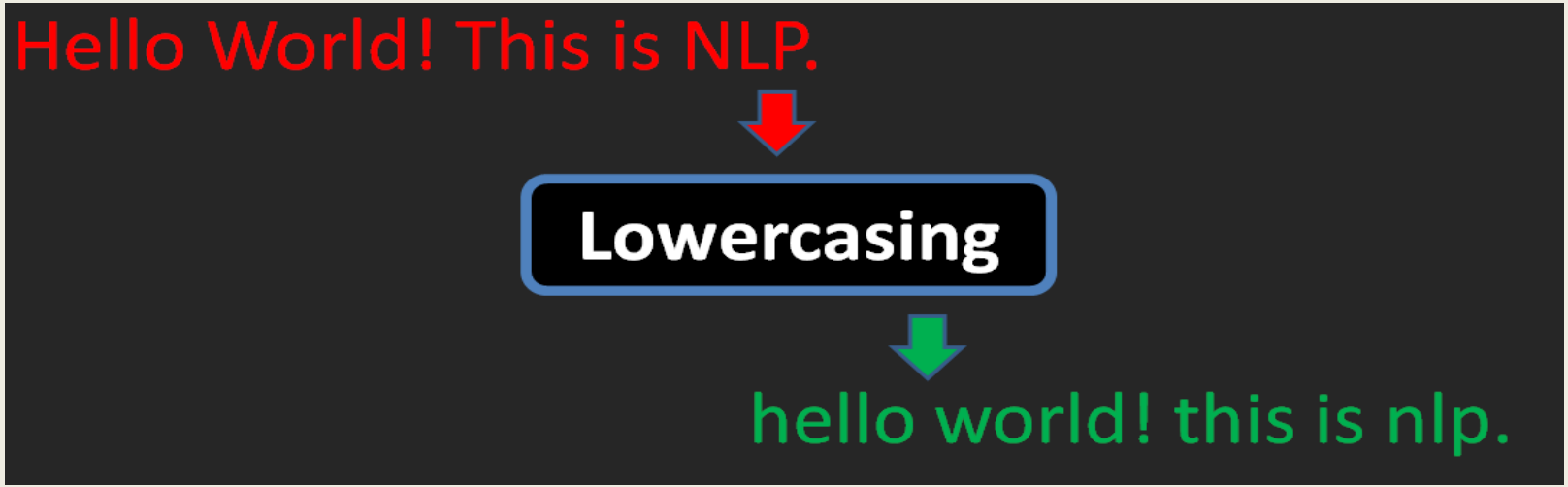
Common NLP Pre-processing step

01. Lowercasing
02. Tokenization
03. Removing Punctuation
04. Removing Stopwords
05. Stemming
06. Lemmatization
07. Removing Numbers
08. Removing Extra Spaces
09. Handling Contractions
10. Removing Special Characters
11. Part-of-Speech (POS) Tagging
12. Named Entity Recognition (NER)
13. Vectorization
14. Handling Missing Data
15. Normalization
16. Spelling Correction
17. Handling Emojis and Emoticons
18. Removing HTML Tags
19. Handling URLs
20. Handling Mentions and Hashtags
21. Sentence Segmentation
22. Handling Abbreviations
23. Language Detection
24. Text Encoding
25. Handling Whitespace Tokens
26. Handling Dates and Times
27. Text Augmentation
28. Handling Negations
29. Dependency Parsing
30. Handling Rare Words
31. Text Chunking
32. Handling Synonyms
33. Text Normalization for Social Media

1. Lowercasing

Purpose: Converts all text to lowercase to ensure uniformity.

Why: Reduces the vocabulary size and avoids treating the same word in different cases as different tokens (e.g., "Apple" vs. "apple").



1. Lowercasing

```
text = "Hello World! This is NLP."
text = text.lower()
print(text)
```

```
hello world! this is nlp.
```

Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

2. Tokenization

Purpose: Splits text into individual words, phrases, or sentences (tokens).

Why: Breaks down text into manageable units for further processing.

2. Tokenization

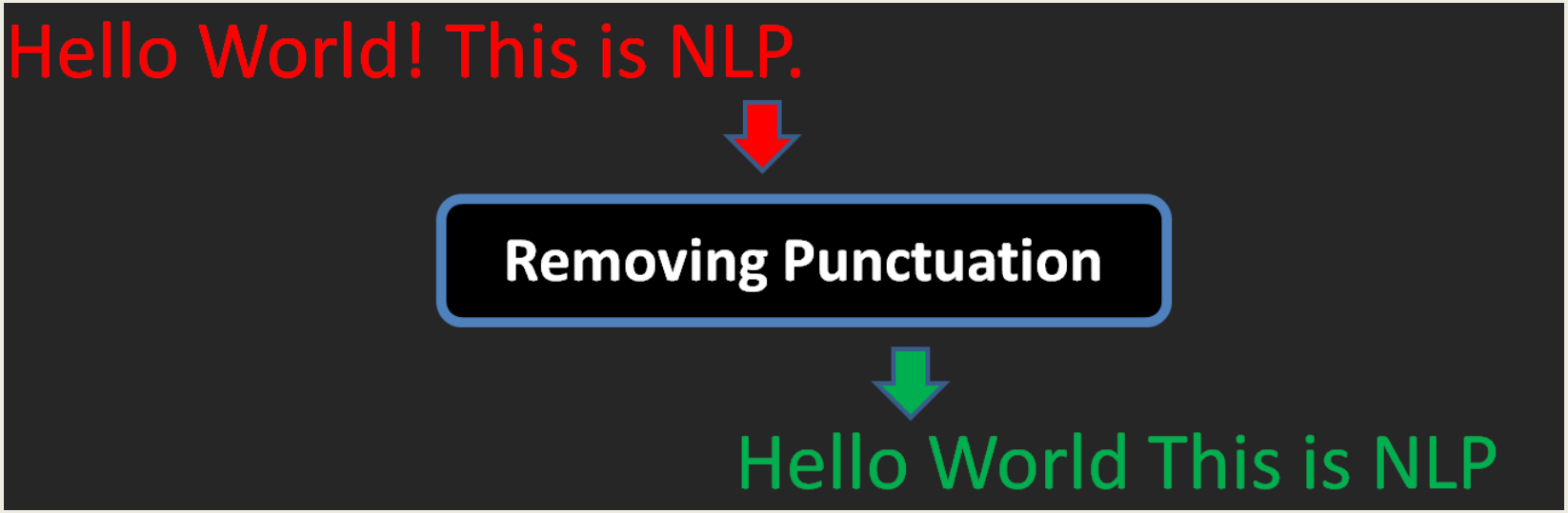
```
import nltk
nltk.download('punkt_tab')
from nltk.tokenize import word_tokenize

text = "Hello World! This is NLP."
tokens = word_tokenize(text)
print(tokens)
```

```
['Hello', 'World', '!', 'This', 'is', 'NLP', '.']
```

3. Removing Punctuation

Purpose: Removes punctuation marks like commas, periods, exclamation marks, etc.

Why: Punctuation often doesn't contribute to the meaning in many NLP tasks and can add noise.



3. Removing Punctuation

```
import string

text = "Hello, World! This is NLP."
text = text.translate(str.maketrans('', '', string.punctuation))
print(text)
```

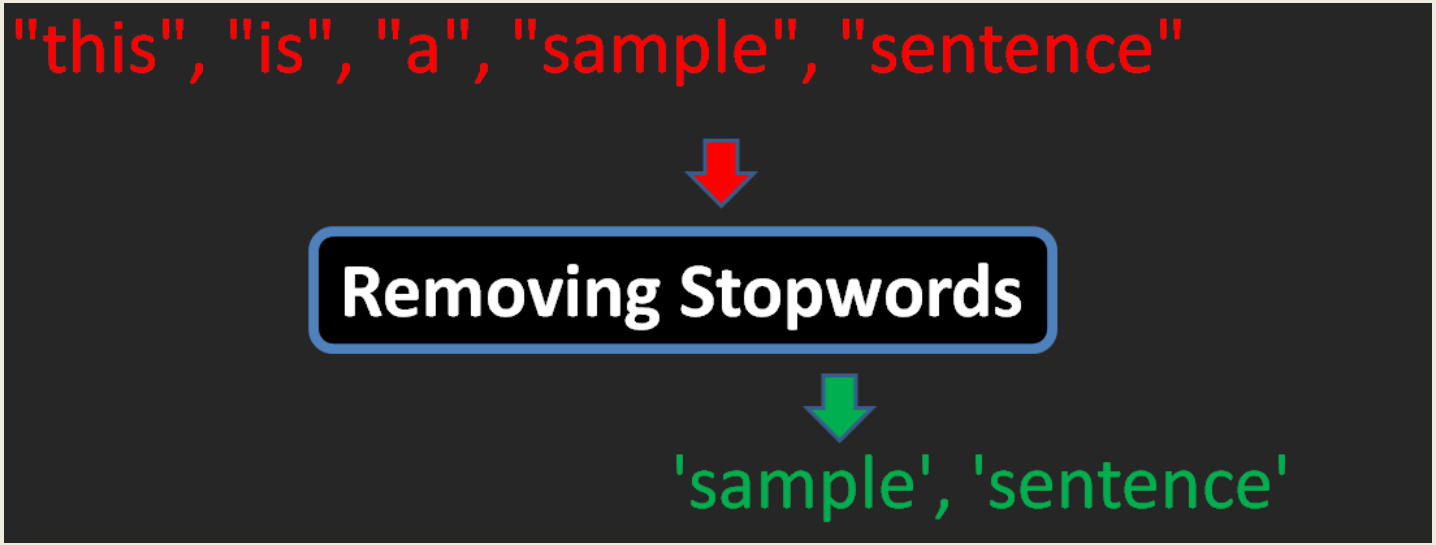
```
Hello World This is NLP
```

Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

4. Removing Stopwords

Purpose: Removes common words like "the," "is," "and," which don't carry significant meaning.

Why: Reduces noise and focuses on meaningful words.



4. Removing Stopwords

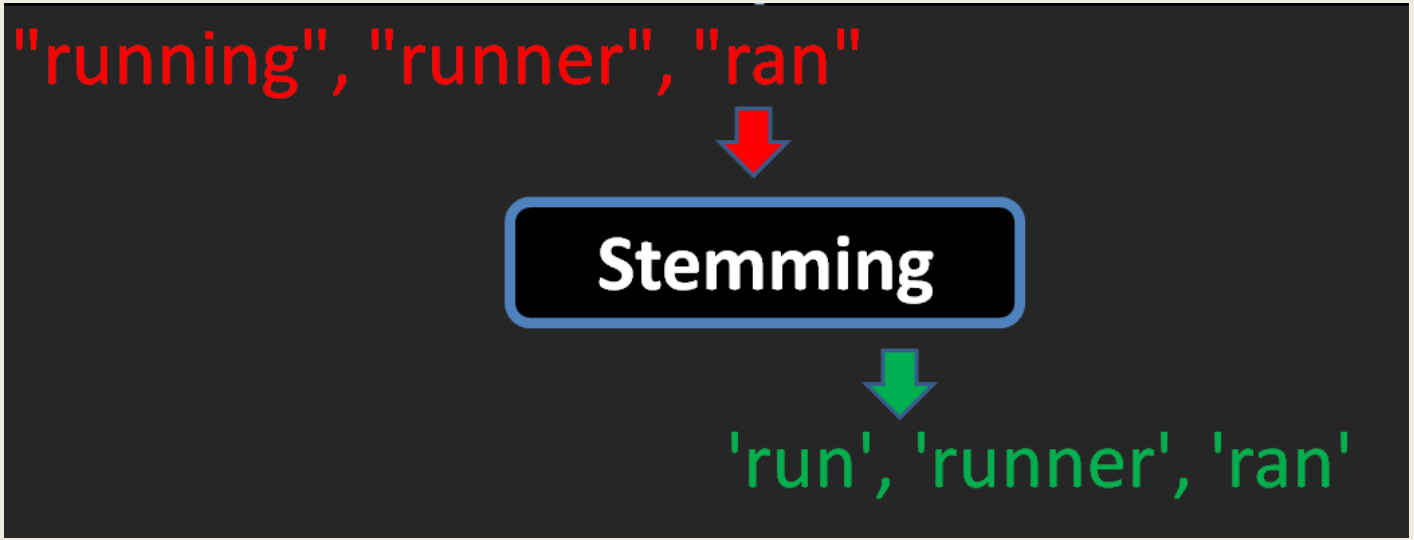
```
import nltk
nltk.download('stopwords') # Download the 'stopwords' dataset
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))

tokens = ["this", "is", "a", "sample", "sentence"]
filtered_tokens = [word for word in tokens if word.lower()
not in stop_words]

print(filtered_tokens)
```

```
['sample', 'sentence']
```

5. Stemming

Purpose: Reduces words to their root form by chopping off suffixes (e.g., "running" → "run").

Why: Simplifies words to their base form, reducing vocabulary size.



5. Stemming

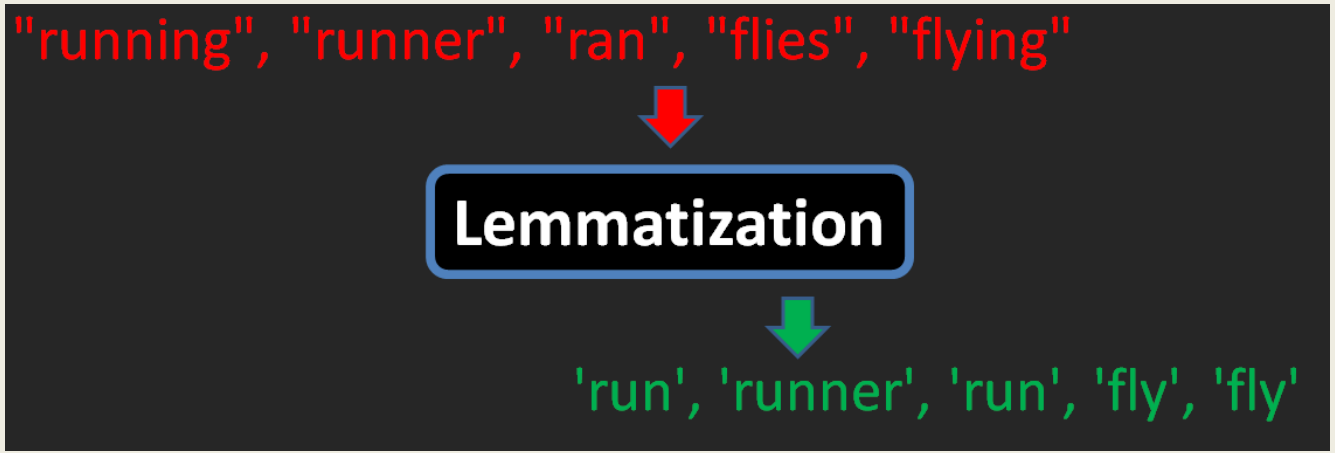
```
from nltk.stem import PorterStemmer


stemmer = PorterStemmer()
words = ["running", "runner", "ran"]
stemmed_words = [stemmer.stem(word) for
word in words]


print(stemmed_words)
```

```
['run', 'runner', 'ran']
```

6. Lemmatization

Purpose: Converts words to their base or dictionary form (e.g., "better" → "good").

Why: More accurate than stemming as it uses vocabulary and morphological analysis.



6. Lemmatization

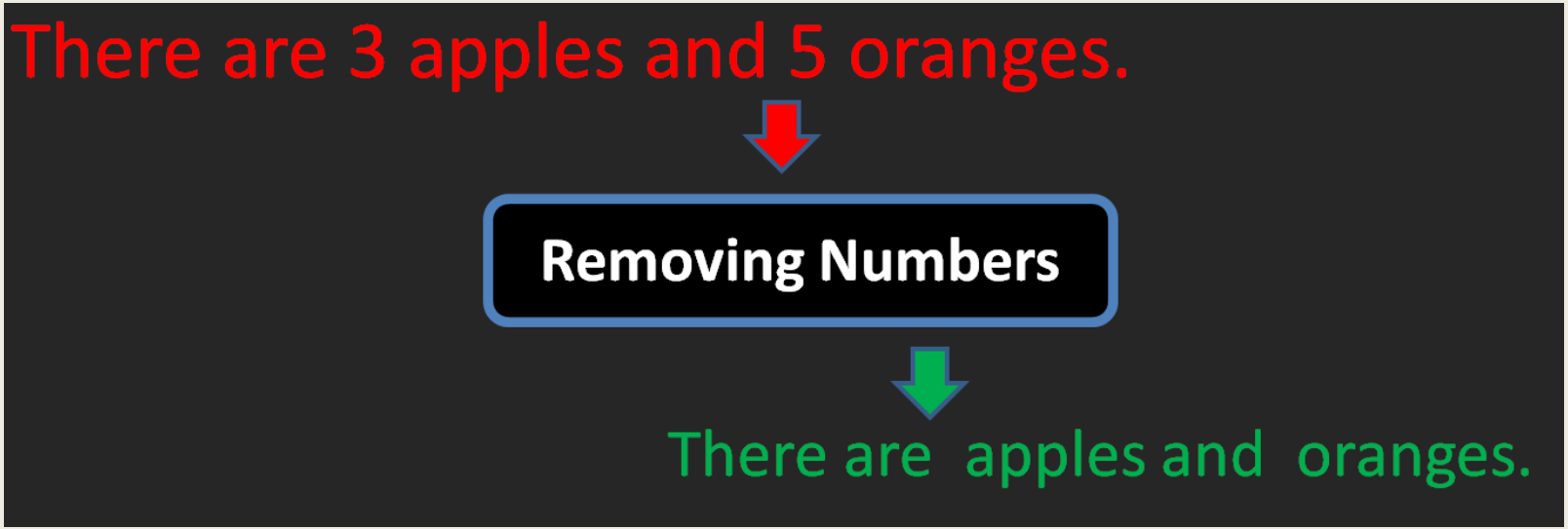
```
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
words = ["running", "runner", "ran", "flies", "flying"]

lemmatized_words = [lemmatizer.lemmatize(word, pos='v') for word in words]

print(lemmatized_words)
```

```
['run', 'runner', 'run', 'fly', 'fly']
```

7. Removing Numbers

Purpose: Removes numeric values from the text.

Why: Numbers may not be relevant in certain NLP tasks like sentiment analysis.



7. Removing Numbers

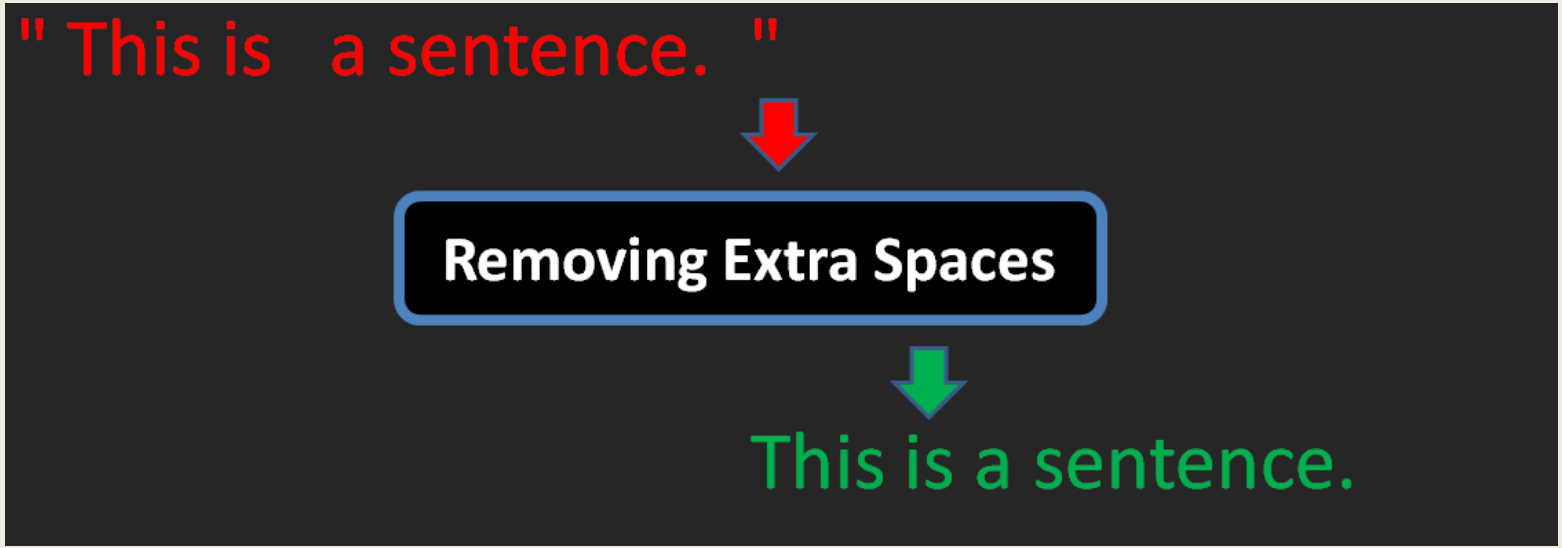
```
import re
text = "There are 3 apples and 5 oranges."
text = re.sub(r'\d+', '', text)
print(text)
```

```
There are  apples and  oranges.
```

8. Removing Extra Spaces

Purpose: Eliminates multiple spaces, tabs, or newlines.

Why: Ensures clean and consistent text formatting.



Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

8. Removing Extra Spaces

```
text = " This is    a sentence.  "
text = ' '.join(text.split())
print(text)
```

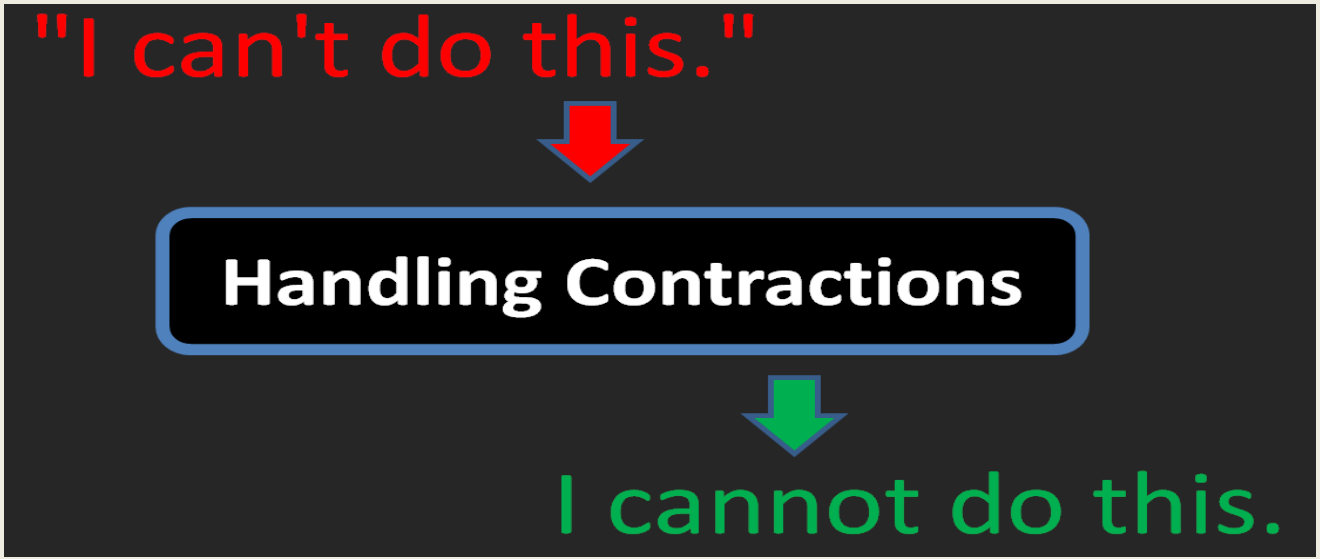
```
This is a sentence.
```

Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

9. Handling Contractions

Purpose: Expands contractions (e.g., "can't" → "cannot").

Why: Standardizes text for better processing.



9. Handling Contractions

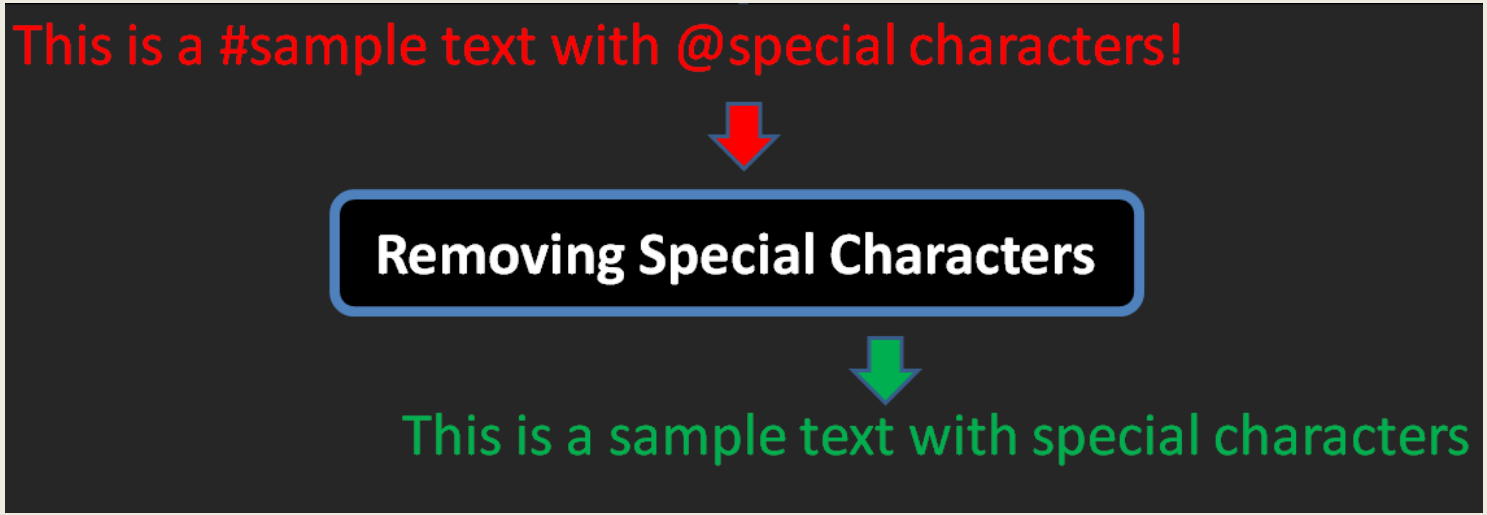
```
!pip install contractions
from contractions import fix
text = "I can't do this."
text = fix(text)
print(text)
```

```
I cannot do this.
```

10. Removing Special Characters

Purpose: Removes non-alphanumeric characters like @, #, $, etc.

Why: Reduces noise and irrelevant symbols.



10. Removing Special Characters

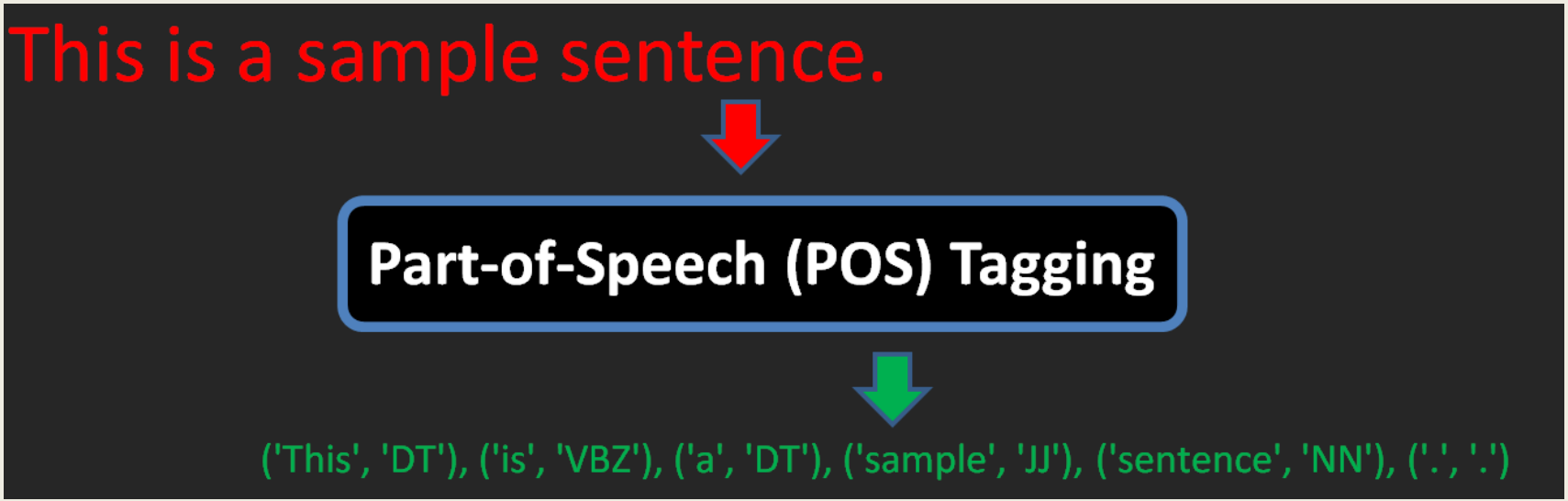
```
import re
text = "This is a #sample text with @special characters!"
text = re.sub(r'[^\w\s]', '', text)
print(text)
```

```
This is a sample text with special characters
```

11. Part-of-Speech (POS) Tagging

Purpose: Assigns grammatical tags to words (e.g., noun, verb, adjective).

Why: Helps in understanding the syntactic structure of sentences.



11. Part-of-Speech (POS) Tagging

Example:

Let's take the sentence: "The cat sat on the mat."

A POS tagger would label each word as follows:

The/DT (Determiner)

cat/NN (Noun)

sat/VBD (Verb, past tense)

on/IN (Preposition)

the/DT (Determiner)

mat/NN (Noun)

Note: Part-of-Speech tagging is a crucial technique in NLP that assigns grammatical labels to words, providing valuable information for various language processing tasks.

11. Part-of-Speech (POS) Tagging

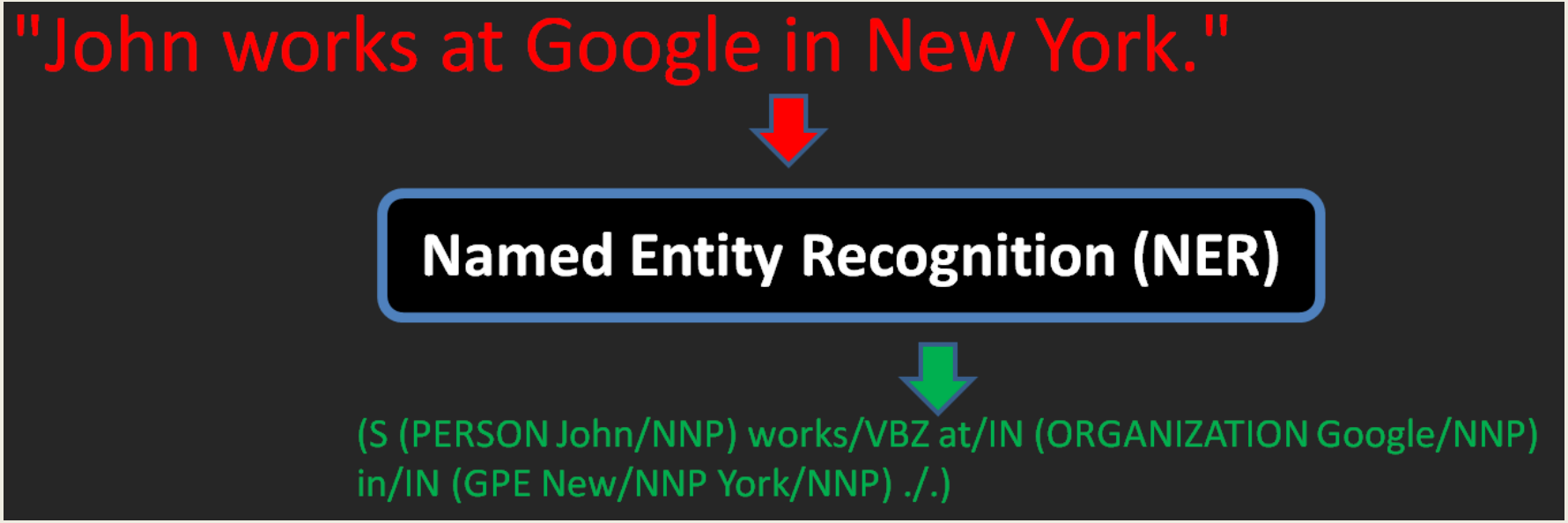
```
import nltk
from nltk import pos_tag
from nltk.tokenize import word_tokenize # Download the required resource

nltk.download('averaged_perceptron_tagger_eng')
tokens = word_tokenize("This is a sample sentence.")
pos_tags = pos_tag(tokens)
print(pos_tags)
```

```
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sample', 'JJ'), ('sentence', 'NN'), ('.', '.')]
```

12. Named Entity Recognition (NER)

Purpose: Identifies and classifies entities like names, dates, locations, etc.

Why: Useful for tasks like information extraction.



12. Named Entity Recognition (NER)

"**Elon Musk**, CEO of **Tesla**, announced on **Twitter** that he will visit **Berlin, Germany** on **July 15th, 2024** to discuss the new **Gigafactory**.“

An NER system would identify:

Elon Musk - **PERSON**
Tesla - **ORGANIZATION**
Twitter - **ORGANIZATION**
Berlin, Germany - **LOCATION**
July 15th, 2024 - **DATE**
Gigafactory - **ORGANIZATION**

Named Entity Recognition

In the 19th century, there was something called the “cult of domesticit
many American women. This meant that most married women were e
stay in the home and raise children. As in other countries, American w
very much under the control of their husband, and had almost no righ
who were not married had only a few jobs open to them, such as wor
clothing factories and serving as maids. By the 19th century, women s
Lucretia Mott and Elizabeth Cady Stanton thought that women should
rights. In 1848, many of these women met and agreed to fight for mo
women, including voting. Many of the women involved in the moveme
women’s rights were also involved in the movement to end slavery.

Tag colors:

12. Named Entity Recognition (NER)

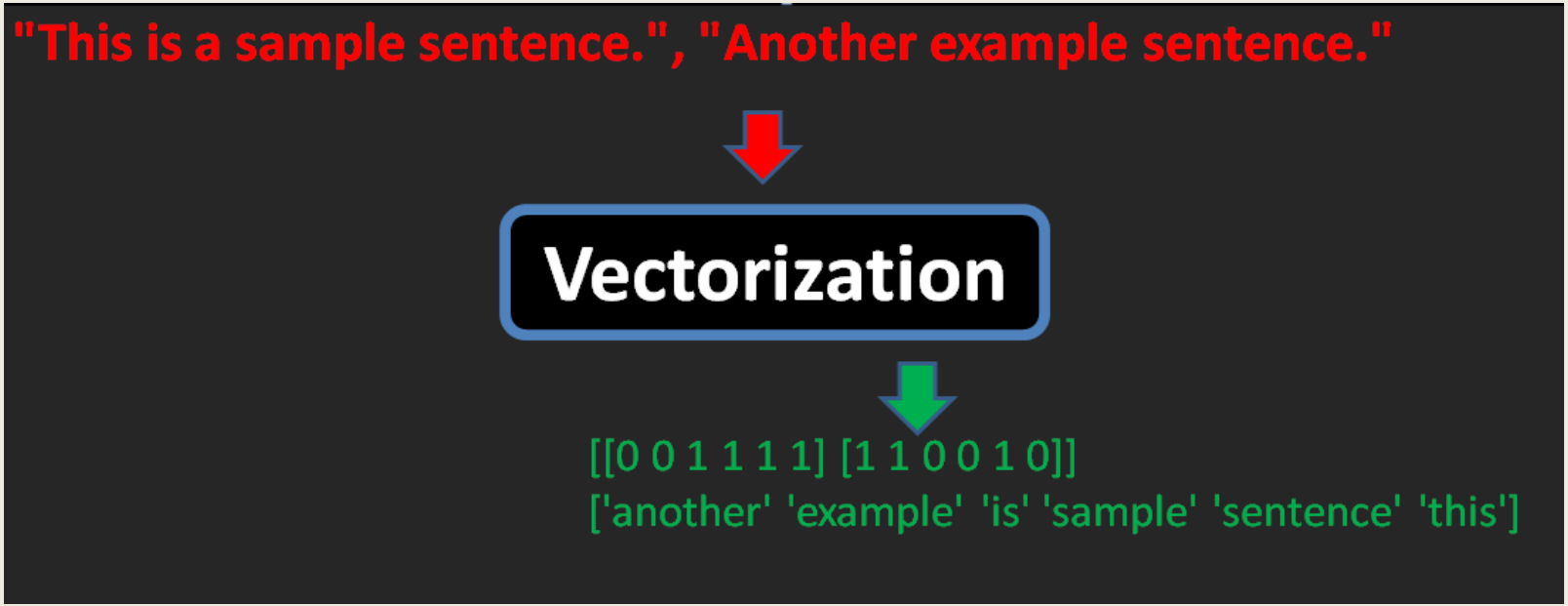
```
import nltk
from nltk import pos_tag, ne_chunk
from nltk.tokenize import word_tokenize # Download the required resources

nltk.download('words')
nltk.download('maxent_ne_chunker')
nltk.download('averaged_perceptron_tagger') # Download the 'maxent_ne_chunker_tab' resource

nltk.download('maxent_ne_chunker_tab') # This line is crucial to fix the error.

tokens = word_tokenize("John works at Google in New York.")
pos_tags = pos_tag(tokens)
ner_tags = ne_chunk(pos_tags)
print(ner_tags)
```

```
(S (PERSON John/NNP) works/VBZ at/IN (ORGANIZATION Google/NNP)
in/IN (GPE New/NNP York/NNP) ./.)
```

13. Vectorization

Purpose: Converts text into numerical vectors (e.g., Bag of Words, TF-IDF, Word Embeddings).

Why: Machine learning models require numerical input.



Vectorization in NLP

In Natural Language Processing (NLP), vectorization is the process of converting text data into numerical representations (vectors) that machine learning models can understand and process. Since computers primarily work with numbers, text data needs to be transformed into a numerical format before it can be used for tasks like sentiment analysis, text classification, or machine translation.

Why is Vectorization Important?

Machine Learning Compatibility: Most machine learning algorithms require numerical input. Text data, in its raw form, is not directly compatible.

Feature Extraction: Vectorization helps extract meaningful features from text, capturing semantic relationships and patterns.

Dimensionality Reduction: It can help reduce the dimensionality of text data, making it more manageable for computation and analysis.

1. Bag of Words (BoW):

Concept: Represents a document as an unordered set of words, disregarding grammar and word order but keeping track of word frequencies.

Process:

- **Vocabulary Creation:** A vocabulary of all unique words across all documents is created.
- **Vector Representation:** Each document is represented as a vector where each element corresponds to a word in the vocabulary. The value of each element indicates the frequency of that word in the document.

Example:

Documents:

Document 1: "The cat sat on the mat."

Document 2: "The dog chased the cat."

Vocabulary: ["the", "cat", "sat", "on", "mat", "dog", "chased"]

Vectors:

Document 1: [2, 1, 1, 1, 1, 0, 0]

Document 2: [1, 1, 0, 0, 0, 1, 1]

Limitations:

Ignores word order and context.

Doesn't capture semantic meaning.

High dimensionality for large vocabularies.

2. Term Frequency-Inverse Document Frequency (TF-IDF):

Concept: Improves upon BoW by weighing words based on their importance in a document relative to the entire corpus.

Process:

- **Term Frequency (TF):** Measures how frequently a term appears in a document.
- **Inverse Document Frequency (IDF):** Measures how rare or common a term is across the entire corpus. Rare terms have higher IDF values.
- **TF-IDF Score:** Calculated as TF * IDF.

Example: (Using the same documents as above)

Let's say "the" appears in many documents, while "chased" appears in only a few. TF-IDF would assign a lower weight to "the" and a higher weight to "chased" in Document 2.

Advantages:

Gives more importance to relevant terms.

Reduces the impact of common words.

4. Document Embeddings (Doc2Vec, Sentence-BERT):

Concept: Extends word embeddings to represent entire documents or sentences as vectors.

Process: Similar to word embeddings but trained to learn representations for larger chunks of text.

Example:

Doc2Vec can create vectors for entire paragraphs, capturing the overall meaning and context.

Advantages:

- Captures document-level semantics.
- Useful for tasks like document similarity and clustering.

Choosing the Right Vectorization Technique:

The best vectorization technique depends on the specific NLP task and dataset.

Simple Tasks: BoW or TF-IDF might be sufficient for tasks like spam detection or basic text classification.

Complex Tasks: Word embeddings and document embeddings are generally preferred for tasks involving semantic understanding, such as sentiment analysis, question answering, and machine translation.

In summary, vectorization is a fundamental step in NLP that allows us to convert text data into a numerical format suitable for machine learning models, enabling computers to understand and process human language.

14. Handling Missing Data

Purpose: Fills or removes missing or incomplete text data.

Why: Ensures the dataset is complete and consistent.

14. Handling Missing Data

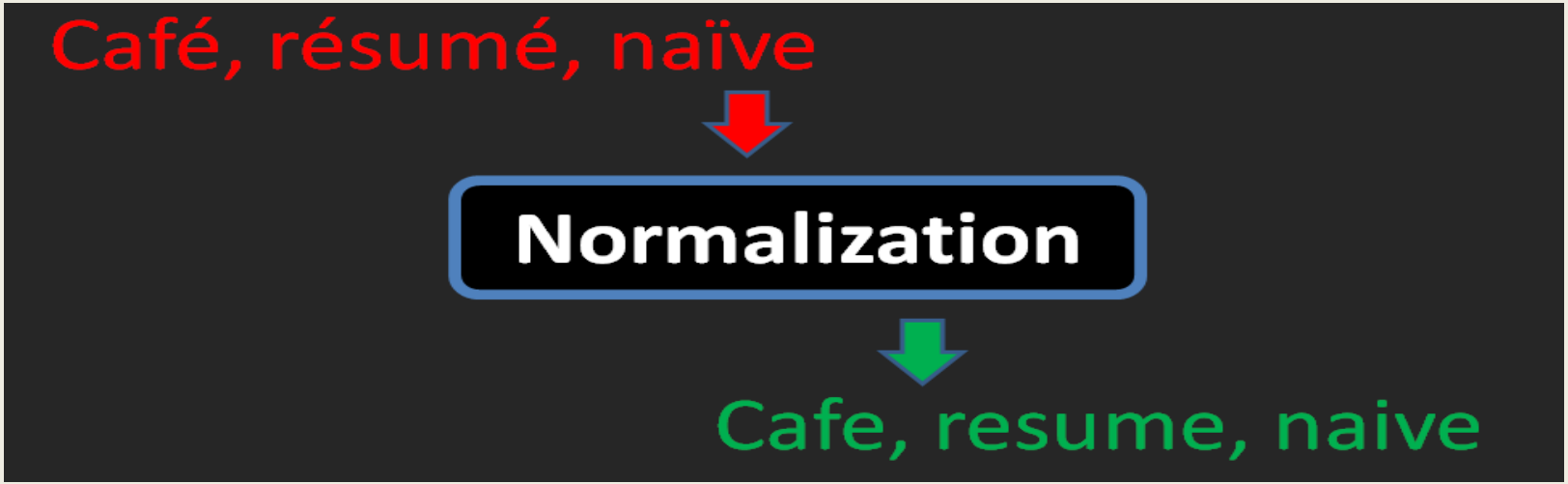
```
import pandas as pd
data = {"text": ["Hello", None, "World"]}
df = pd.DataFrame(data)
df["text"].fillna("My Dear", inplace=True) # Fill missing values
print(df)
```

```
      text
0    Hello
1  My Dear
2    World
```

15. Normalization

Purpose: Standardizes text (e.g., converting all dates to a single format).

Why: Ensures consistency in the dataset.



Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

15. Normalization

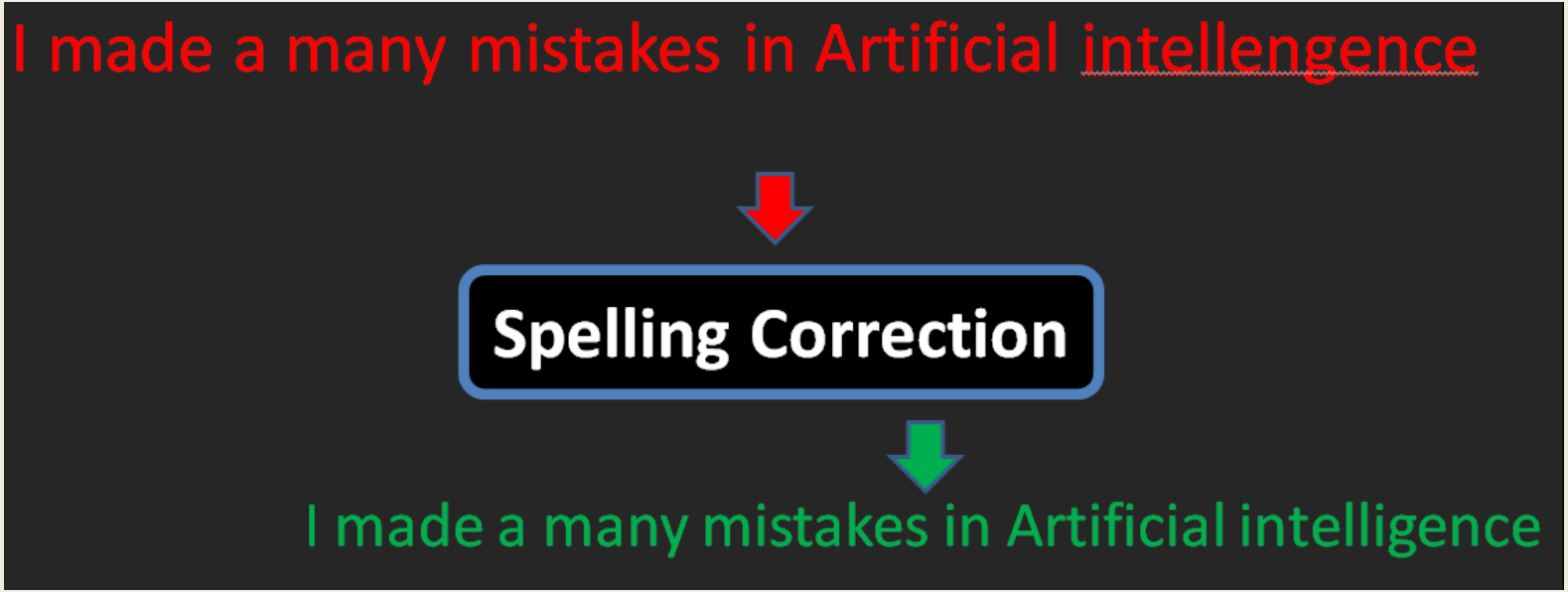
```
import unicodedata
text = "Café, résumé, naïve"
text = unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode('utf-8')
print(text)
```

```
Cafe, resume, naive
```

16. Spelling Correction

Purpose: Corrects spelling errors in the text.

Why: Improves the quality of the text for analysis.



16. Spelling Correction

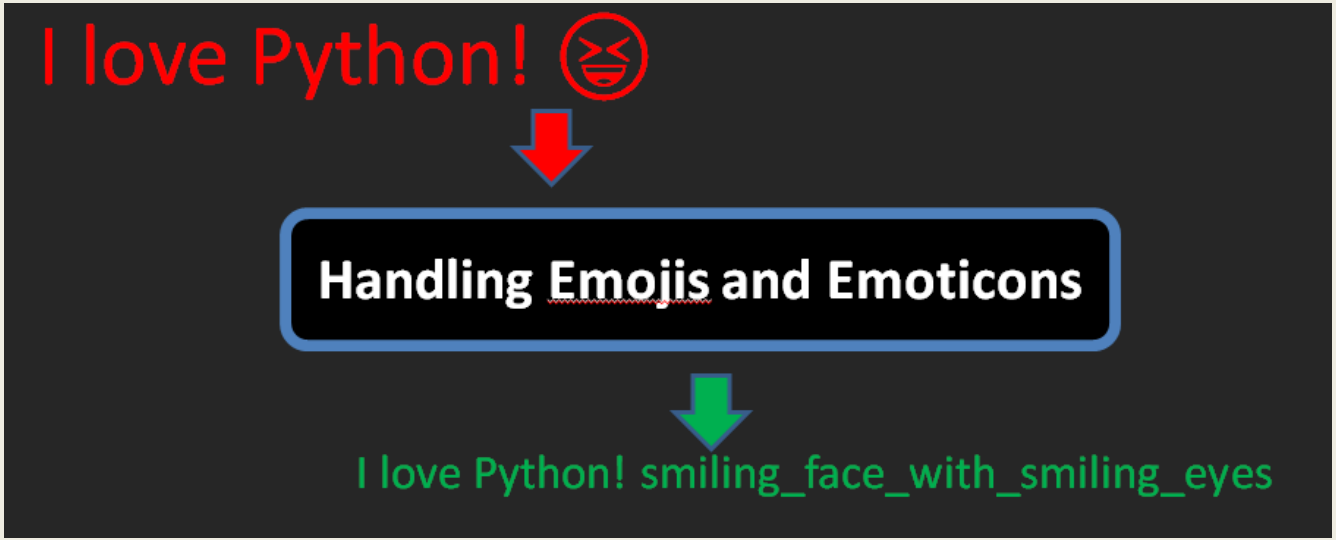
```
from textblob import TextBlob
text = "I made a many mistakes in Artificial intellengence"
blob = TextBlob(text)
corrected_text = blob.correct()
print(corrected_text)
```

```
I made a many mistakes in Artificial intelligence
```

17. Handling Emojis and Emoticons

Purpose: Converts emojis and emoticons into text or removes them.

Why: Emojis can carry sentiment or meaning that needs to be captured.



17. Handling Emojis and Emoticons

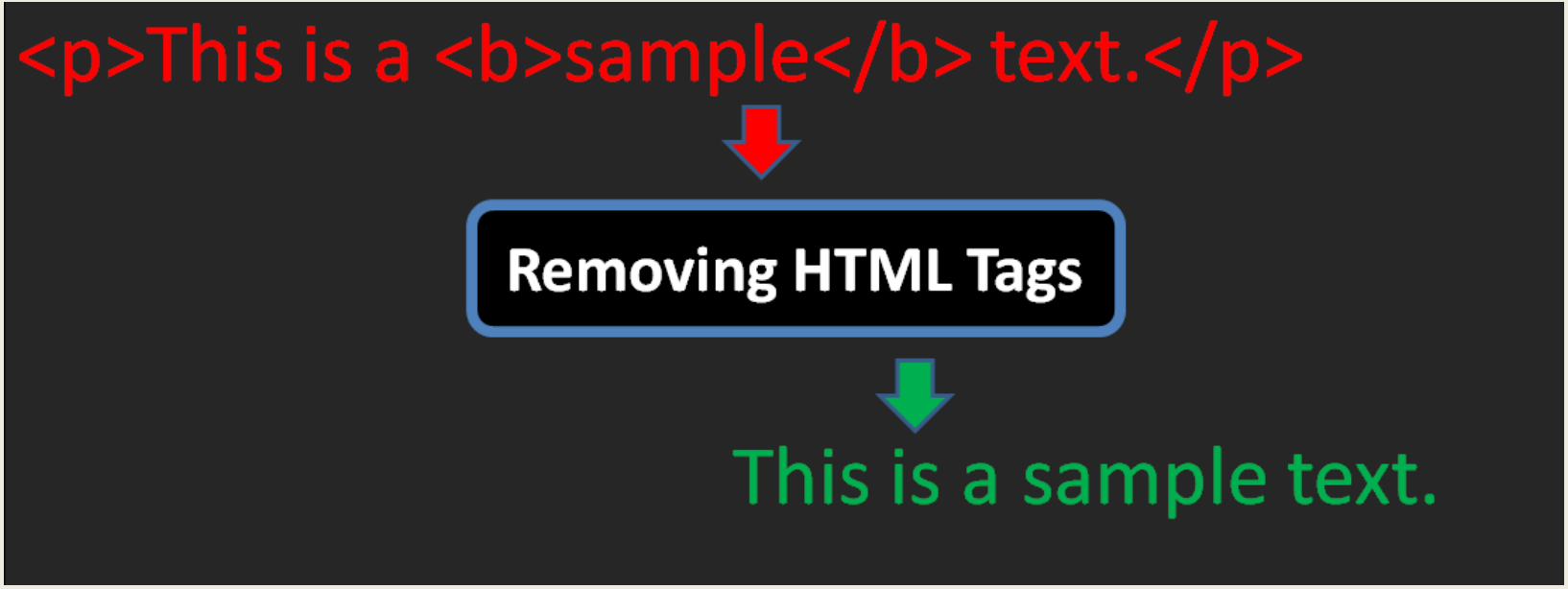
```
!pip install emoji
import emoji text = "I love Python! 😊 " # Convert emojis to text
text = emoji.demojize(text)
print(text) # Output: "I love Python! :smiling_face_with_smiling_eyes:"
# Remove emojis
text = emoji.replace_emoji(text, replace="")
print(text)
```

I love Python! smiling_face_with_smiling_eyes

18. Removing HTML Tags

Purpose: Removes HTML tags from web scraped text.

Why: HTML tags are irrelevant for most NLP tasks.



18. Removing HTML Tags

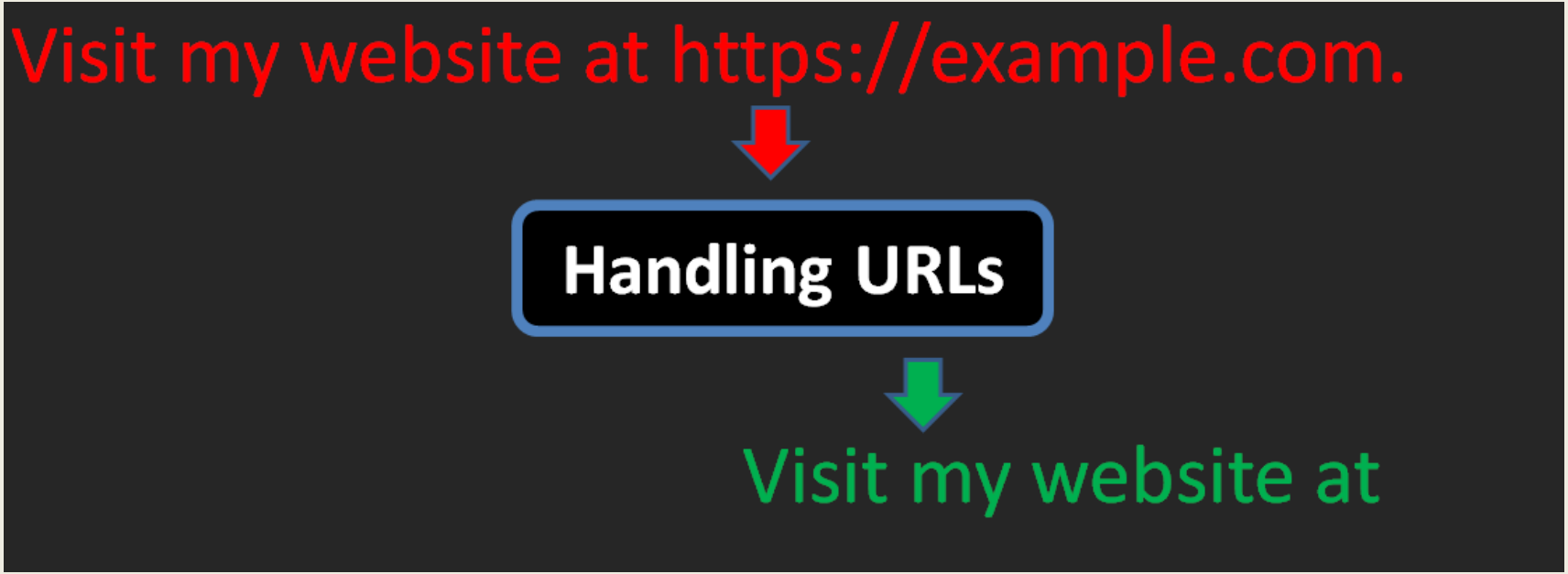
```
from bs4 import BeautifulSoup
text = "<p>This is a <b>sample</b> text.</p>"
soup = BeautifulSoup(text, "html.parser")
clean_text = soup.get_text()
print(clean_text)
```

```
This is a sample text.
```

19. Handling URLs

Purpose: Removes or replaces URLs in the text.

Why: URLs are often irrelevant for text analysis.



19. Handling URLs

```
import re
text = "Visit my website at https://example.com."
text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
print(text)
```

```
Visit my website at
```

20. Handling Mentions and Hashtags

Purpose: Processes or removes social media mentions (@user) and hashtags (#topic).

Why: Useful for social media text analysis.

Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

20. Handling Mentions and Hashtags

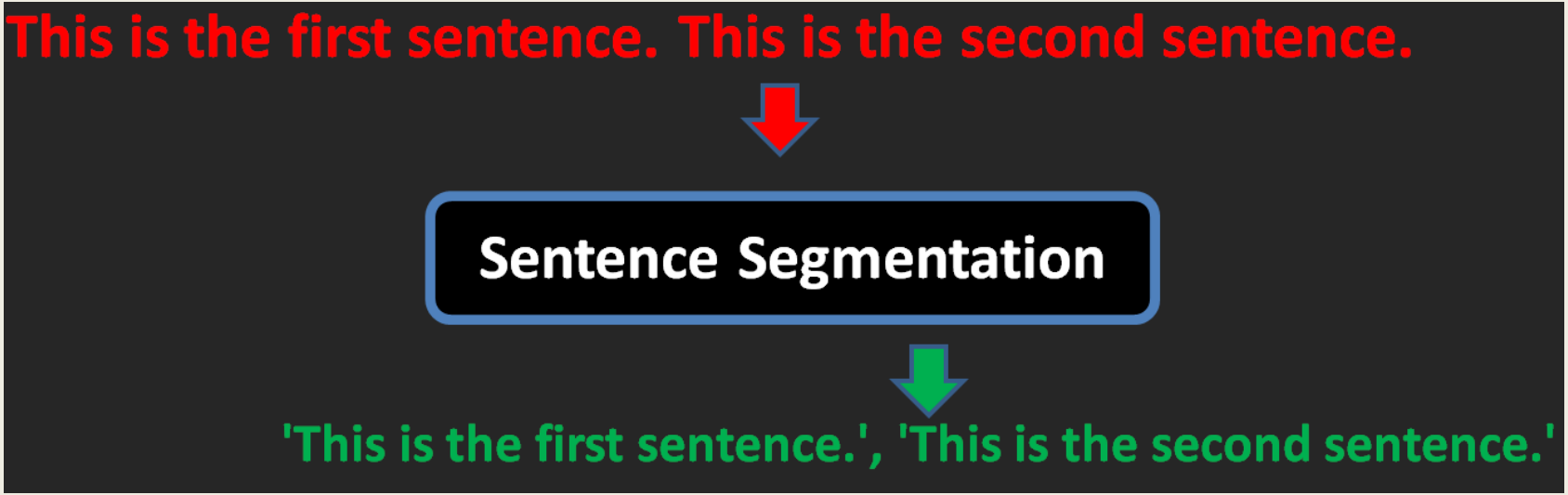
```
import re
text = "Hey @user, check out #NLP!"
text = re.sub(r'@\w+|#\w+', '', text)
print(text)
```

```
Hey , check out !
```

21. Sentence Segmentation

Purpose: Splits text into individual sentences.

Why: Important for tasks like machine translation or summarization.



21. Sentence Segmentation

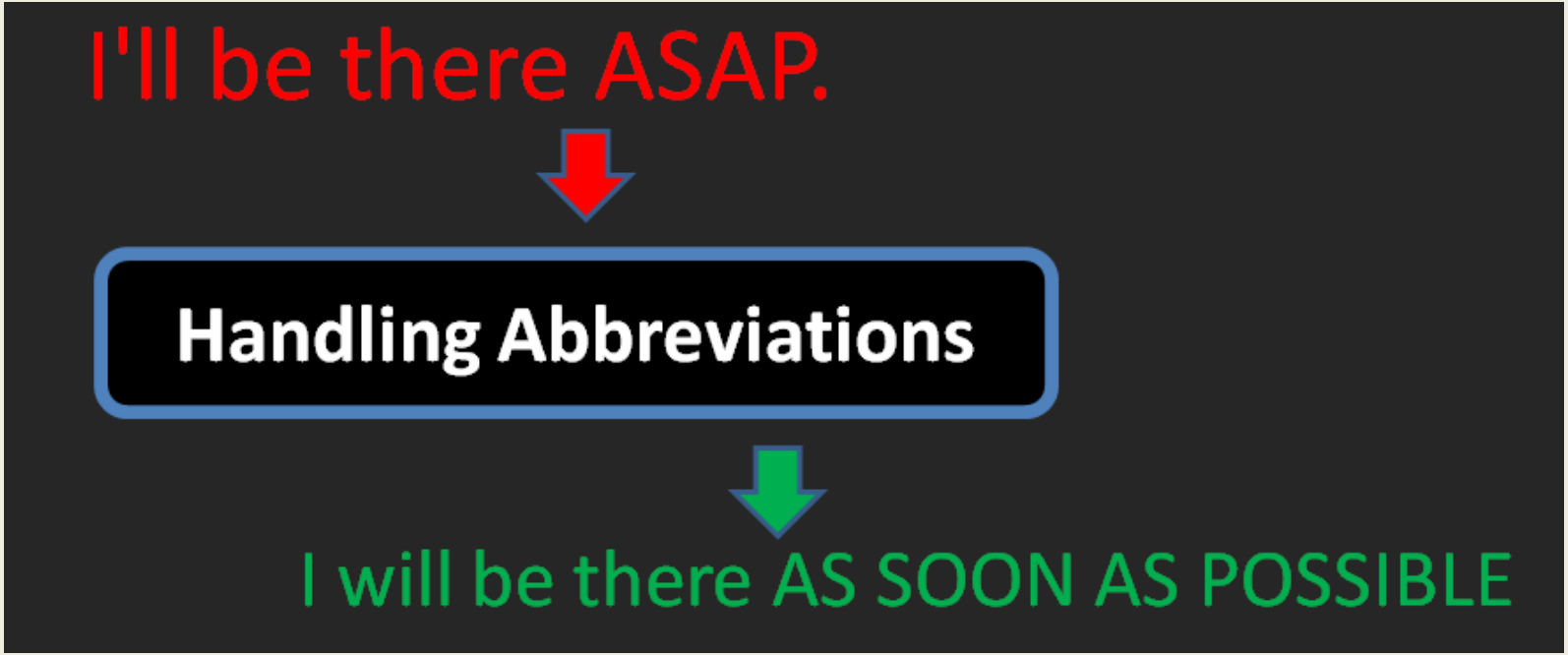
```
import nltk
from nltk.tokenize import sent_tokenize

# Download the 'punkt_tab' dataset before using sent_tokenize
nltk.download('punkt_tab') # This line is crucial to fix the error.

text = "This is the first sentence. This is the second sentence."
sentences = sent_tokenize(text)
print(sentences)
```

```
['This is the first sentence.', 'This is the second sentence.']
```

22. Handling Abbreviations

Purpose: Expands abbreviations (e.g., "ASAP" → "as soon as possible").

Why: Ensures clarity and consistency.



22. Handling Abbreviations

```
!pip install contractions
import contractions
text = "I'll be there ASAP."
expanded_text = contractions.fix(text)
print(expanded_text)
```

```
I will be there AS SOON AS POSSIBLE.
```

23. Language Detection

Purpose: Identifies the language of the text.

Why: Ensures the correct NLP model is applied.

Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

23. Language Detection

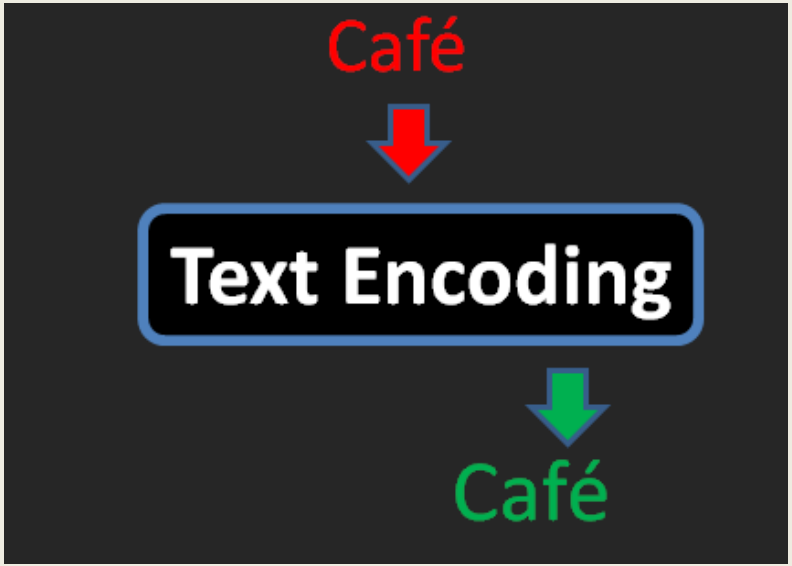
```
!pip install langdetect
from langdetect import detect
text = "Ceci est un texte en français."
language = detect(text)
print(language)
```

```
fr
```

24. Text Encoding

Purpose: Converts text into a specific encoding format (e.g., UTF-8).

Why: Ensures compatibility with NLP tools and models.



24. Text Encoding

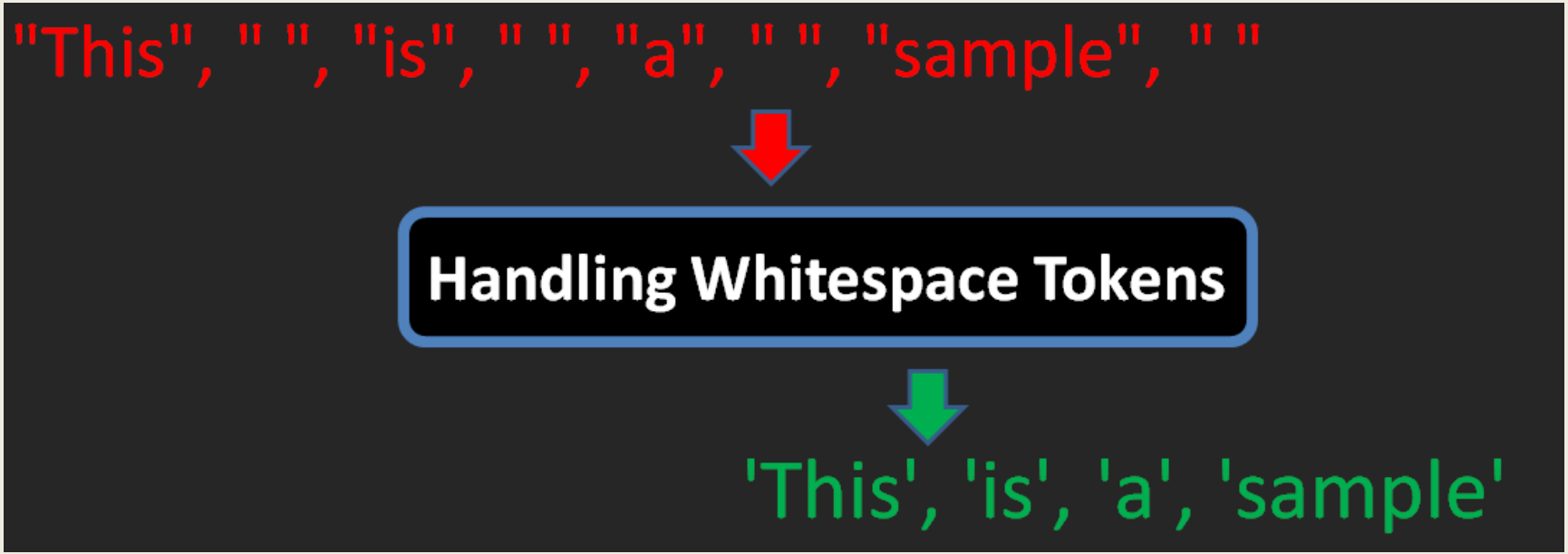
```
text = "Café"
text = text.encode('utf-8').decode('utf-8')
print(text)
```

```
Café
```

25. Handling Whitespace Tokens

Purpose: Removes or processes tokens that are just spaces or empty strings.

Why: Ensures clean and meaningful tokens.



25. Handling Whitespace Tokens

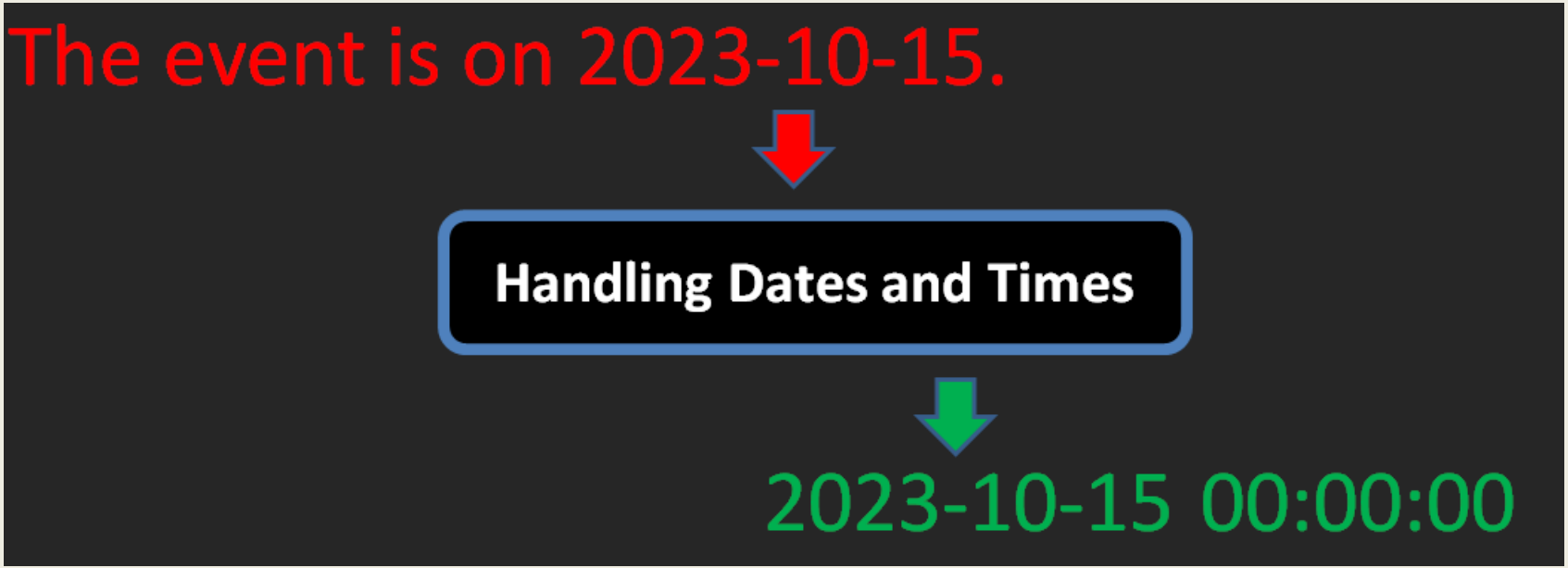
```
tokens = ["This", " ", "is", " ", "a", " ", "sample", " "]
tokens = [token for token in tokens if token.strip()]
print(tokens)
```

```
['This', 'is', 'a', 'sample']
```

26. Handling Dates and Times

Purpose: Standardizes or extracts date and time formats.

Why: Useful for time-sensitive analysis.



26. Handling Dates and Times

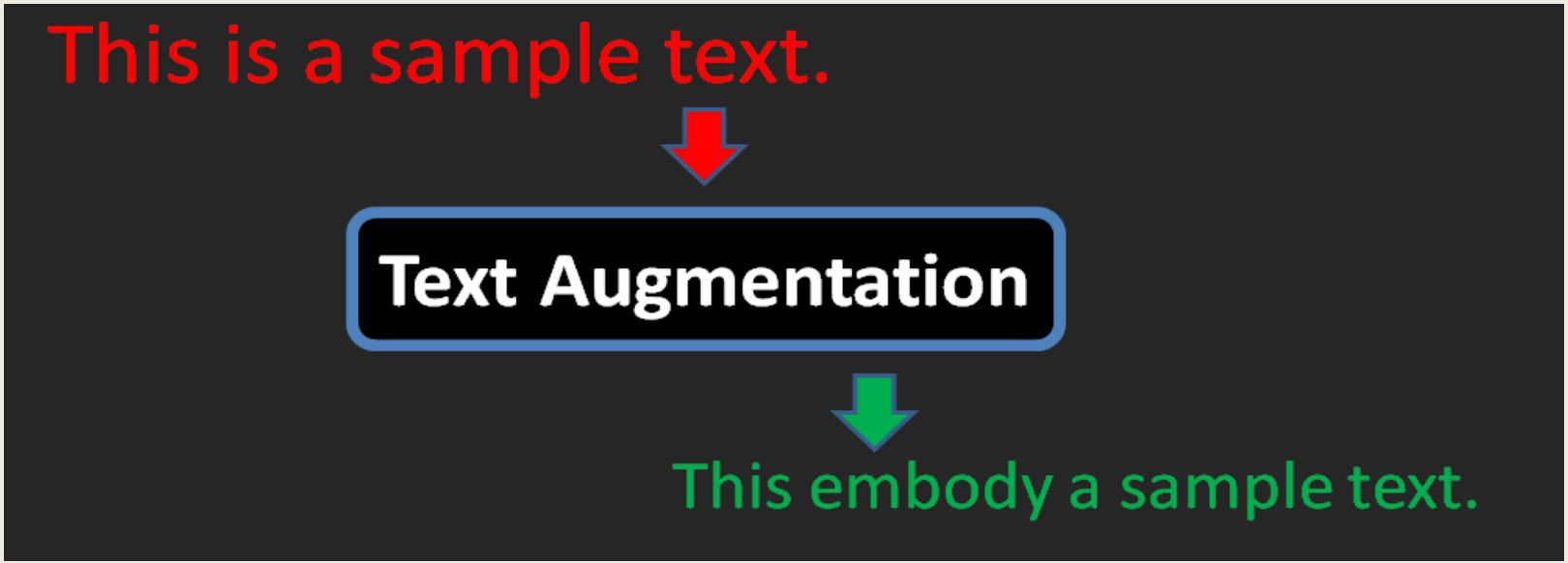
```
import dateutil.parser as dparser
text = "The event is on 2023-10-15."
date = dparser.parse(text, fuzzy=True)
print(date)
```

```
2023-10-15 00:00:00
```

27. Text Augmentation

Purpose: Generates additional training data by modifying existing text (e.g., synonym replacement).

Why: Improves model robustness and performance.



27. Text Augmentation

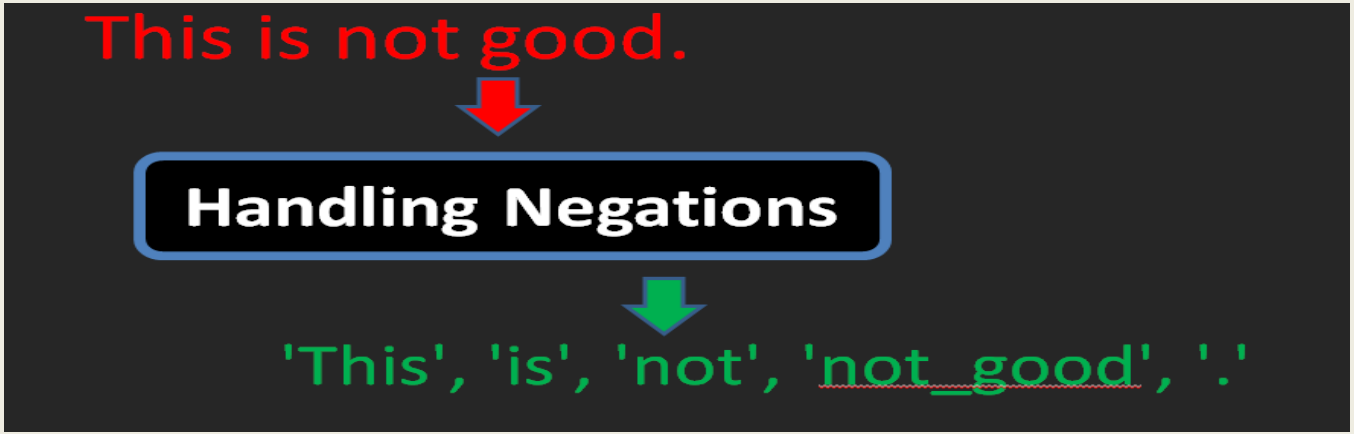
```
!pip install nlpaug
import nltk
from nlpaug.augmenter.word import SynonymAug

# Download the 'averaged_perceptron_tagger_eng' dataset
nltk.download('averaged_perceptron_tagger_eng')

aug = SynonymAug(aug_src='wordnet')
text = "This is a sample text."
augmented_text = aug.augment(text)
print(augmented_text)
```

```
['This embody a sample text.']
```

28. Handling Negations

Purpose: Identifies and processes negations (e.g., "not good").

Why: Important for sentiment analysis and understanding context.



Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

28. Handling Negations

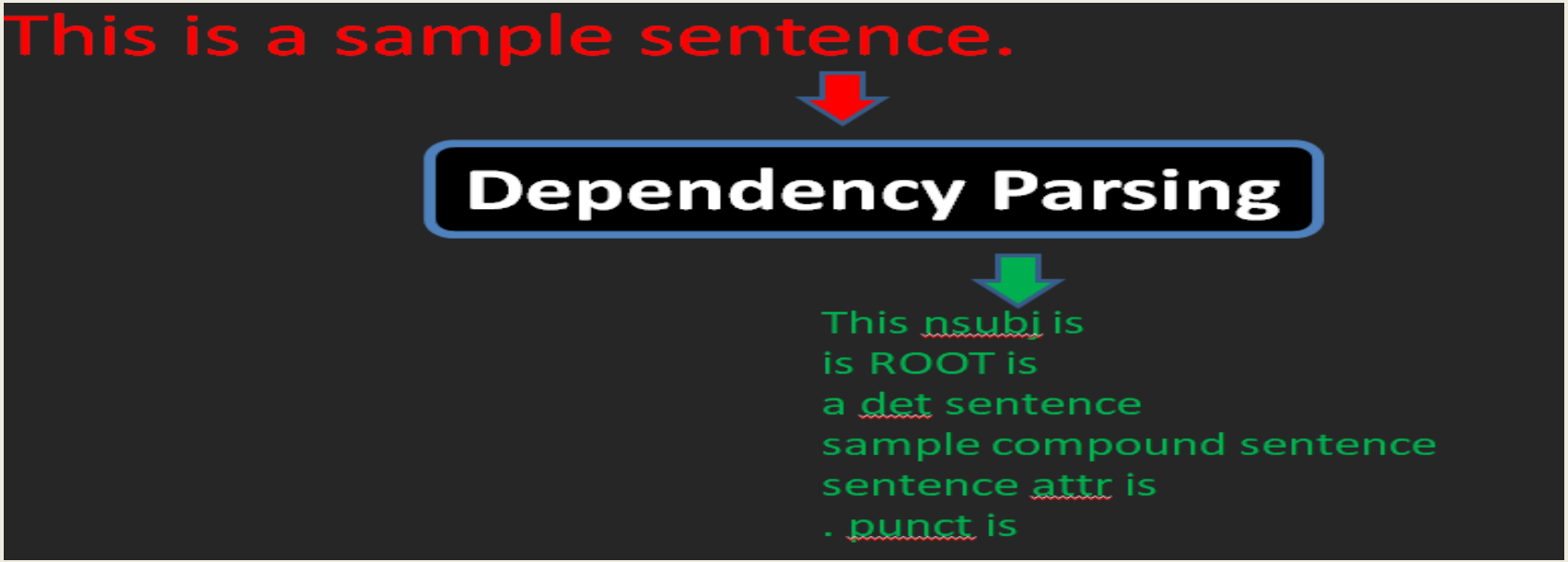
```
from nltk import word_tokenize
text = "This is not good."
tokens = word_tokenize(text)
for i, token in enumerate(tokens):
  if token == "not" and i + 1 < len(tokens):
    tokens[i + 1] = "not_" + tokens[i + 1]
print(tokens)
```

```
['This', 'is', 'not', 'not_good', '.']
```

29. Dependency Parsing

Purpose: Analyzes the grammatical structure of a sentence.

Why: Helps in understanding relationships between words.



29. Dependency Parsing

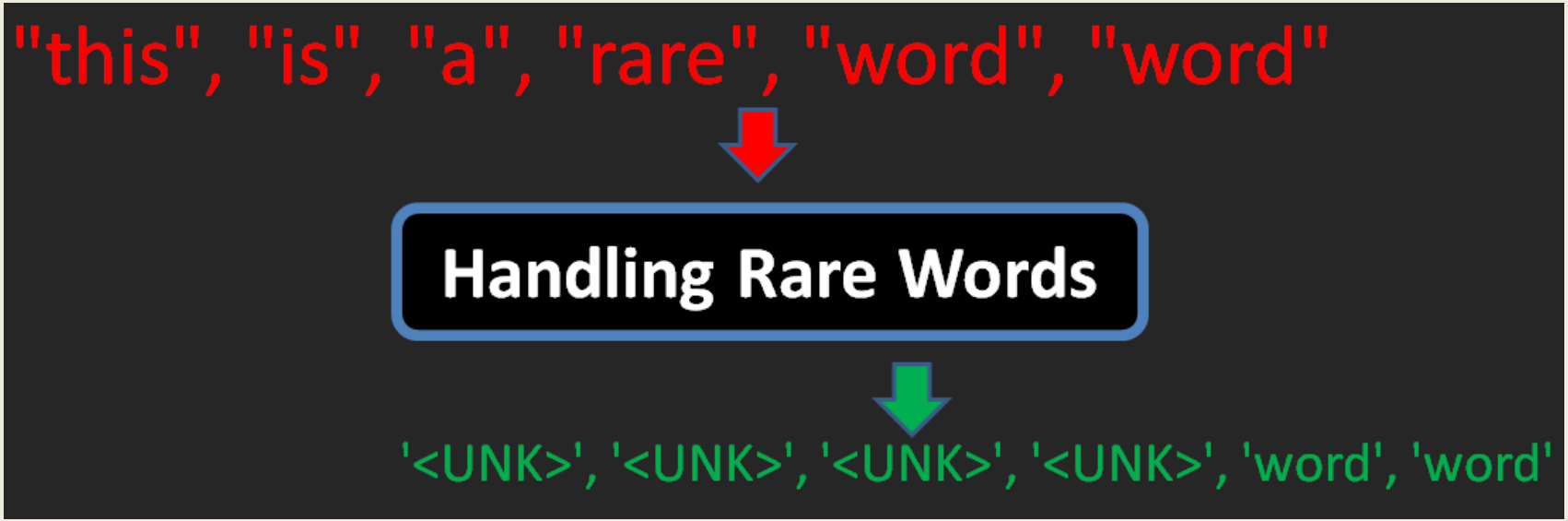
```
import spacy
!python -m spacy download en_core_web_sm # Download the model if not already downloaded
nlp = spacy.load("en_core_web_sm") # Load the model directly using spacy.load
# The rest of your code remains the same
text = "This is a sample sentence."
doc = nlp(text)
for token in doc:
  print(token.text, token.dep_, token.head.text)
```

```
This nsubj is
is ROOT is
a det sentence
sample compound sentence
sentence attr is
. punct is
```

30. Handling Rare Words

Purpose: Replaces or removes rare words that occur infrequently.

Why: Reduces noise and improves model efficiency.



30. Handling Rare Words

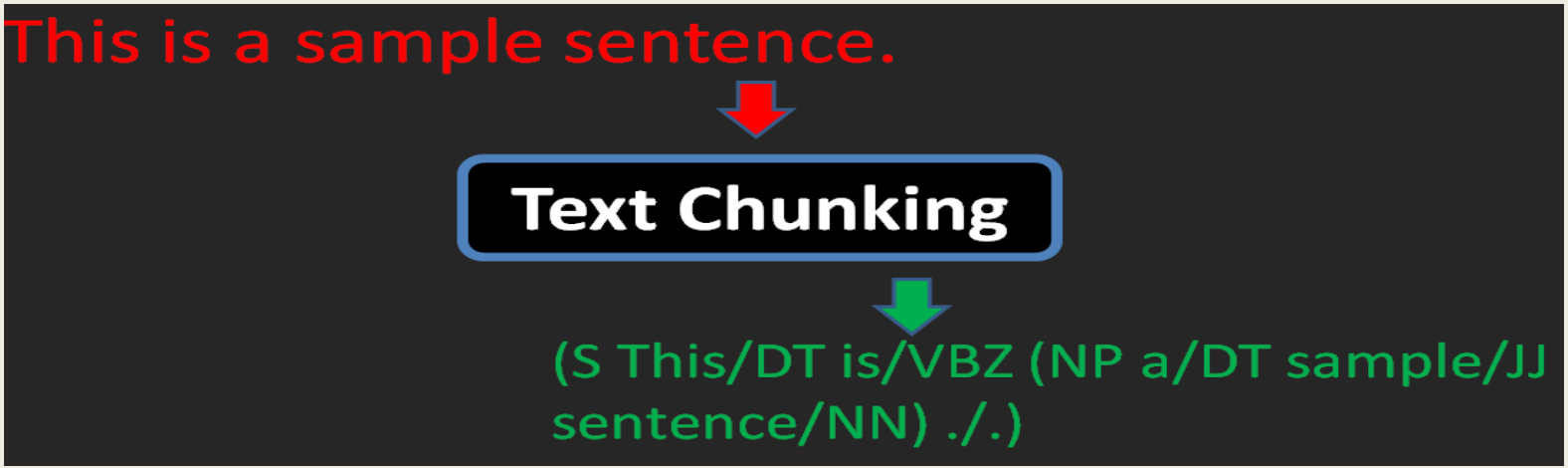
```
from collections import Counter
tokens = ["this", "is", "a", "rare", "word", "word"]
word_counts = Counter(tokens)
rare_words = {word for word, count in word_counts.items() if count < 2}
tokens = [token if token not in rare_words else "<UNK>" for token in tokens]
print(tokens)
```

```
['<UNK>', '<UNK>', '<UNK>', '<UNK>', 'word', 'word']
```

31. Text Chunking

Purpose: Groups words into "chunks" based on POS tags (e.g., noun phrases).

Why: Useful for information extraction.



31. Text Chunking

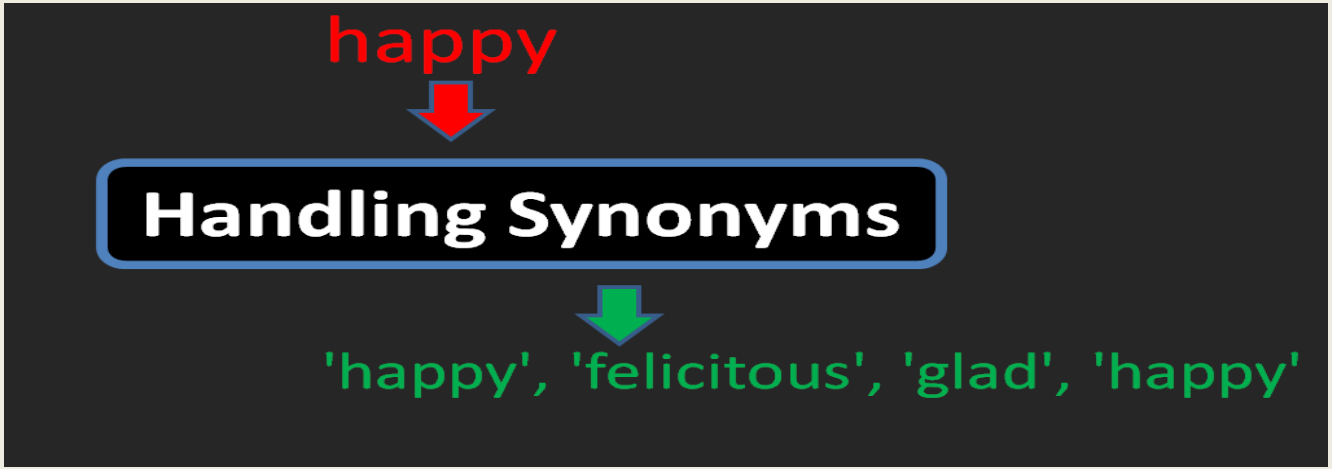
```
from nltk import pos_tag, word_tokenize
from nltk.chunk import RegexpParser
text = "This is a sample sentence."
tokens = word_tokenize(text)
pos_tags = pos_tag(tokens)
grammar = "NP: {<DT>?<JJ>*<NN>}"
chunk_parser = RegexpParser(grammar)
tree = chunk_parser.parse(pos_tags)
print(tree)
```

```
(S This/DT is/VBZ (NP a/DT sample/JJ sentence/NN) ./.)
```

32. Handling Synonyms

Purpose: Replaces words with their synonyms.

Why: Helps in text augmentation and reducing redundancy.



Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

32. Handling Synonyms

```
from nltk.corpus import wordnet
word = "happy"
synonyms = wordnet.synsets(word)
print([syn.lemmas()[0].name() for
syn in synonyms])
```

```
['happy', 'felicitous', 'glad', 'happy']
```

33. Text Normalization for Social Media

Purpose: Processes informal text (e.g., "u" → "you", "gr8" → "great").

Why: Social media text often contains informal language and slang.

33. Text Normalization for Social Media

```
import re
text = "I loooove this!"
text = re.sub(r'(.)\1+', r'\1', text)
print(text)
```

```
I love this!
```

Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/

NLP Preprocessing

These pre-processing steps are crucial for cleaning, standardizing, and transforming raw text into a format suitable for NLP models. The specific steps used depend on the task (e.g., sentiment analysis, machine translation) and the nature of the text (e.g., formal documents, social media posts).

Prepared by: Syed Afroz Ali
Learn NLP Step by Step: https://t.me/AIMLDeepThaught/832
Follow me on LinkedIn for more Content: https://www.linkedin.com/in/syed-afroz-70939914/