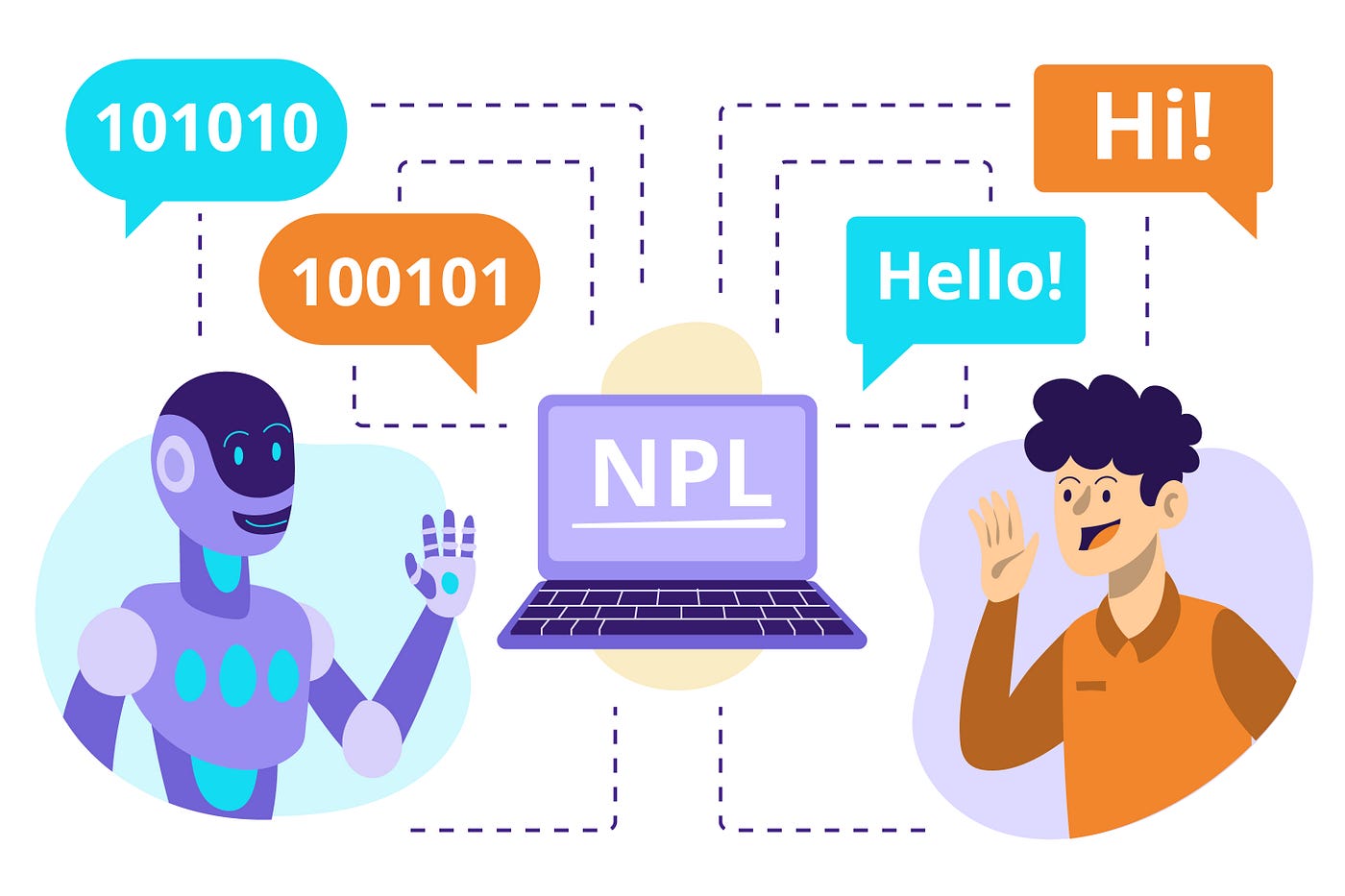
Fundamentals of NLP:Language Structure and Preprocessing Techniques

# Overview of Natural Language Processing (NLP)

## 1. Introduction to NLP

* **What is NLP?**

NLP is a field of artificial intelligence that focuses on the interaction between computers and human language. It involves the development of algorithms and models that enable machines to understand, interpret, and generate human language.

****

* **NLP Libraries:**

## Regex (Regular Expressions) Library - Regex is a very effective tool for Pattern matching and text modification.

1. Regex libraries provide powerful tools for text manipulation and pattern matching in NLP. They enable efficient **preprocessing, data validation, and feature extraction**, making them essential for tasks such as **text cleaning, tokenization, and information extraction.** Using regex effectively can significantly enhance the quality and structure of text data, leading to better results in NLP applications.

Sample code:

import re

text = "Contact us at support@example.com or visit our website at www.example.com. The price is $199.99!”

tokens = re.findall(r'\b\w+\b', text)  # Extracting words

print("Tokens:", tokens)

cleaned\_text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters

print("Cleaned Text:", cleaned\_text)

emails = re.findall(r'[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}', text)

print("Email Addresses:", emails)

prices = re.findall(r'\$\d+(?:\.\d{2})?', text)

print("Prices:", prices)

**output:**

Tokens: ['Contact', 'us', 'at', 'support', 'example', 'com', 'or', 'visit', 'our', 'website', 'at', 'www', 'example', 'com', 'The', 'price', 'is', '199', '99']

Cleaned Text: Contact us at supportexamplecom or visit our website at wwwexamplecom The price is

Email Addresses: ['[support@example.com](mailto:support@example.com)']

Prices: ['$199.99']

## NLTK (Natural Language Toolkit) - which is a powerful toolkit for working with human language data in Python.

**Sample code:**

import nltk

# Download the required resources

nltk.download('punkt')  # For tokenization

nltk.download('averaged\_perceptron\_tagger')  #

import nltk

#Sample text

text = "NLTK is a powerful library for natural language processing."

tokens = nltk.word\_tokenize(text)

pos\_tags = nltk.pos\_tag(tokens)

print("Tokens:", tokens)

print("POS Tags:", pos\_tags)

**OUTPUT:**

Tokens: ['NLTK', 'is', 'a', 'powerful', 'library', 'for', 'natural', 'language', 'processing', '.']

POS Tags: [('NLTK', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('powerful', 'JJ'), ('library', 'NN'), ('for', 'IN'), ('natural', 'JJ'), ('lanuage', 'NN'), ('processing', 'NN'), ('.', '.')]

**Expansion for the output**

* **NNP**: Proper noun, singular (e.g., NLTK)
* **VBZ**: Verb, 3rd person singular present (e.g., is)
* **DT**: Determiner (e.g., a)
* **JJ**: Adjective (e.g., powerful, natural)
* **NN**: Noun, singular or mass (e.g., library, language, processing)
* **IN**: Preposition or subordinating conjunction (e.g., for)
* **.**: Sentence-final punctuation (e.g., .)
* **Applications of NLP :**
* **Text Classification:** Sentiment analysis, spam detection.
* **Machine Translation:** Translating text from one language to another.
* **Chatbots and Virtual Assistants:** Siri, Alexa, Google Assistant.
* **Speech Recognition:** Converting spoken language into text.

## 2. Language Structure

* Language structure in NLP involves understanding how language is constructed, which can be broken down into the following components:
* **Phonology**:
* **Phonetics**: The study of sounds in human speech. In NLP, phonetic representation can help in tasks like speech recognition and text-to-speech synthesis.
* **Phonology**: The study of how sounds function in particular languages, including patterns and rules.
* **Morphology**:

* **Definition**: The study of the structure of words, including roots, prefixes, and suffixes.
* **Morphological Analysis**: Involves identifying the base forms of words (lemmatization) and their grammatical forms (inflection).
* **Example**: The word "running" can be broken down into its root "run" and the suffix "-ing.
* **Syntax**:

 **Definition**: The set of rules that govern the structure of sentences.

 **Parse Trees**: Used to represent the syntactic structure of

sentences, showing the relationships between words.

 **Grammar Types**:

* **Constituency Grammar**: Focuses on phrases and their hierarchical structure.
* **Dependency Grammar**: Focuses on relationships between words (e.g., subject-verb relationships).
* **Lexical Semantics**
* **Word Relationships**: Understanding relationships between words, such as synonyms, antonyms, and hypernyms (hierarchical relationships).
* **Word Embeddings**: Representing words in a continuous vector space, capturing semantic relationships (e.g., Word2Vec, GloVe).
* **Semantics**:
  + **Definition**: The study of meaning in language.
* **Word Sense Disambiguation**: Determining the correct meaning of a word based on context.
* **Semantic Role Labeling**: Identifying the roles that words play in the context of a sentence (e.g., who did what).
* **Pragmatics**:

 **Definition**: The study of how context influences the interpretation of

meaning.

 **Contextual Understanding**: Analyzing language use in specific

situations, including idioms, sarcasm, and inferred meanings.

## 3. Preprocessing Techniques

* Preprocessing is a critical step in NLP that involves cleaning and transforming raw text into a format suitable for modeling. Common preprocessing techniques include:
* **Tokenization**: Splitting text into individual words or tokens.

Eg. String = “ The World is very Beautiful” , String = [“The / world / is / Very /

Beautiful”

Code:

import nltk

from nltk.tokenize

import word\_tokenize

String = “The World is Beautiful”

word\_tokensize(String) # code for word\_tokensize

output :

['The', 'world', 'is', 'very', 'beautiful']

Code :

from nltk.tokenize import sent\_tokenize

String = “The World is Beautiful. The world is mine”

sent\_tokenize(String) # code for sentence\_tokensize

output :

['The world is very beautiful.', 'hello world']

**Lowercasing**: Converting all text to lowercase to ensure uniformity.

CODE:

import nltk

from nltk.tokenize import word\_tokenize

String = "The World is Beautiful"

lowercase\_string = String.lower()

print("Lowercase String:", lowercase\_string)

OUTPUT:

Lowercase String: the world is beautiful

* **Stopword Removal**: Removing common words (like "and," "the," "is") that may not contribute significant meaning.

Code:

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

tokenized\_words = ['i', 'am', 'going', 'to', 'go', 'to', 'the', 'store', 'and', 'park']

stop\_words = stopwords.words('english')

print(stop\_words)

filtered\_words = [word for word in tokenized\_words if word not in stop\_words]

print("Filtered Words:", filtered\_words)

print("First 5 Stopwords:", stop\_words[:5])

output :

['going', 'go', 'store', 'park']

**['i', 'me', 'my', 'myself', 'we']**

* **Stemming**: Reducing words to their root form (e.g., "running" to "run").

Code :

import nltk

from nltk.stem import PorterStemmer

nltk.download('punkt')  # This is necessary for tokenization, not PorterStemmer itself

tokenized\_words = ['i', 'am', 'humbled', 'by', 'this', 'traditional', 'meeting']

porter = PorterStemmer()

stemmed\_words = [porter.stem(word) for word in tokenized\_words]

print("Stemmed Words:", stemmed\_words)

output :

['i', 'am', 'humbl', 'by', 'thi', 'tradit', 'meet']

* **Cleaning Text**: Clean the text data in the method of strip, replace and split

Code :

import nltk

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

nltk.download('punkt')

nltk.download('stopwords')

text = [" The India Story. by Bimal Jalal ", " The London Adventure. by Ruskin Bond "]

strip\_whitespace = [word\_tokenize(string.strip()) for string in text]

remove\_periods = [[word.lower() for word in tokens if word.isalpha()] for tokens in strip\_whitespace]

stop\_words = set(stopwords.words('english'))

filtered\_text = [[word for word in tokens if word not in stop\_words] for tokens in remove\_periods]

print("Tokens after stripping whitespace:", strip\_whitespace)

print("Tokens after removing periods and lowercasing:", remove\_periods)

print("Tokens after removing stop words:", filtered\_text)

output :

Tokens after stripping whitespace: [['The', 'India', 'Story', '.', 'by', 'Bimal', 'Jalal'], ['The', 'London', 'Adventure', '.', 'by', 'Ruskin', 'Bond']]

Tokens after removing periods and lowercasing: [['the', 'india', 'story', 'by', 'bimal', 'jalal'], ['the', 'london', 'adventure', 'by', 'ruskin', 'bond']]

Tokens after removing stop words: [['india', 'story', 'bimal', 'jalal'], ['london', 'adventure', 'ruskin', 'bond']]

* **Lemmatization**: Similar to stemming but more sophisticated, considering the context to reduce words to their base form.

Code :

from nltk.stem import WordNetLemmatizer

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

nltk.download('wordnet')

text = "Natural Language Processing (NLP) is a fascinating field of AI!"

lemmatizer = WordNetLemmatizer()

tokenized\_text = word\_tokenize(text)

lemmas = [lemmatizer.lemmatize(word.lower()) for word in tokenized\_text if word.isalpha()]

print("Lemmas:", lemmas)

**output :**

**['natural', 'language', 'processing', 'nlp', 'fascinating', 'field', 'ai']**

* **Removing Punctuation**: Eliminating punctuation marks to reduce noise.

Code :

import nltk

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

text = "Natural Language Processing (NLP) is a fascinating field of AI!"

tokens = word\_tokenize(text)

tokens = [word for word in tokens if word.isalpha()]

print("Tokens:", tokens)

**output:x**

Tokens: ['Natural', 'Language', 'Processing', 'NLP', 'is', 'a', 'fascinating', 'field', 'of', 'AI'

* **Text Normalization**: Converting numbers, abbreviations, and other non-standard forms to a standard form.
* **Remove Numbers :** Remove numbers if they are not relevant to the text analysis.

**Code :**

import re

text = "There are 123 apples."

normalized\_text = re.sub(r'\d+', '', text)

print("Normalized Text:", normalized\_text)

**output :**

**There are apples.**

* **Expanding Contractions :** Convert contractions into their full forms.

Eg. Can’t – Cannot.

Code :

!pip install contractions

from contractions import fix

text = "I can't do this."

expanded\_text = fix(text)

print("Expanded Text:", expanded\_text)

output :

I cannot do this.

* **Normalization of Slang/Abbreviations** : Convert slang or abbreviations to their standard forms. Eg. U – You

Code :

import nltk

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

# Your slang dictionary

slang\_dict = {"brb": "be right back", "u": "you"}

# Text to normalize

text = "brb, I'll talk to u later."

# Tokenize the text

tokens = word\_tokenize(text)

# Normalize tokens using the slang dictionary

normalized\_tokens = [slang\_dict.get(word, word) for word in tokens]

# Join the normalized tokens back into a string

normalized\_text = " ".join(normalized\_tokens)

print("Normalized Text:", normalized\_text)

output :

be right back , I 'll talk to you later .

* **Handling Accented Characters :** Convert accented characters to their non-accented equivalents. Eg. Café – Cafe

**Code :**

!pip install unidecode

import unidecode

text = "más"

normalized\_text = unidecode.unidecode(text)

print(normalized\_text)

output :

mas

4. **Hands-On Lab with Text Preprocessing in Python :**

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import PorterStemmer, WordNetLemmatizer

import string

# Sample text

text = "Natural Language Processing (NLP) is a fascinating field of AI!"

# Lowercasing

text = text.lower()

# Tokenization

tokens = word\_tokenize(text)

# Removing Punctuation

tokens = [word for word in tokens if word.isalpha()]

# Stopword Removal

stop\_words = set(stopwords.words('english'))

tokens = [word for word in tokens if word not in stop\_words]

# Stemming

stemmer = PorterStemmer()

stems = [stemmer.stem(word) for word in tokens]

# Lemmatization

lemmatizer = WordNetLemmatizer()

lemmas = [lemmatizer.lemmatize(word) for word in tokens]

print("Original Text:", text)

print("Tokens:", tokens)

print("Stems:", stems)

print("Lemmas:", lemmas)

output :

Original Text: natural language processing (nlp) is a fascinating field of ai!

Tokens: ['natural', 'language', 'processing', 'nlp', 'fascinating', 'field', 'ai']

Stems: ['natur', 'languag', 'process', 'nlp', 'fascin', 'field', 'ai']

Lemmas: ['natural', 'language', 'processing', 'nlp', 'fascinating', 'field', 'ai']