cloudthat

Natural Language Processing

Natural Language Processing

By “**natural language**” we mean a language that is used for everyday communication by humans.

NLP is an Intersection of several fields

- Computer Science
- Artificial Intelligence
- Linguistics

It is basically teaching computers to process human language

Two main components:

- Natural Language Understanding (NLU)
- Natural Language Generation (NLG)

Natural Language Processing

The goal of NLP is to enable machines to understand, interpret, and generate human language in a way that is both meaningful and contextually relevant. This interdisciplinary field involves aspects of computer science, linguistics, and cognitive psychology.

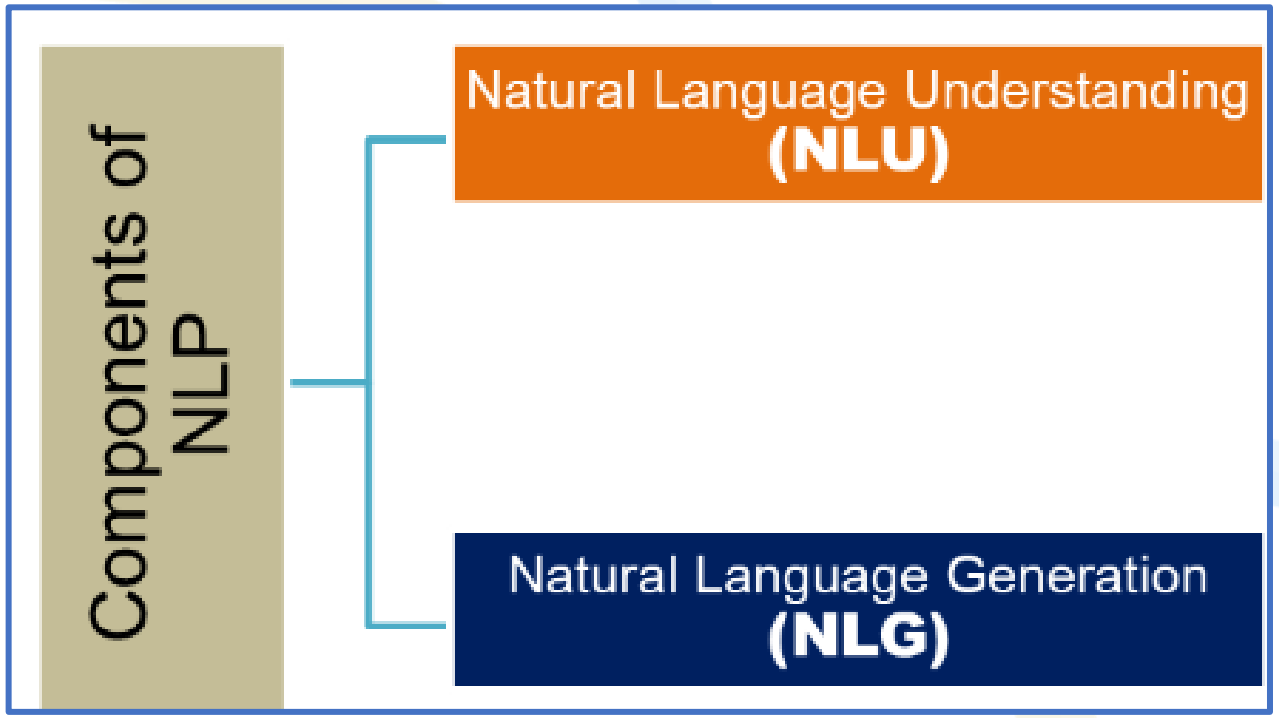
Here are some key components and tasks within NLP:

- Text Understanding
- Syntax and Semantics
- Language Generation
- Machine Translation
- Sentiment Analysis
- Information Extraction
- Question Answering
- Speech Recognition
- Speech Synthesyis

Components of NLP

The two basic components in which NLP can be divided are as follows:

- Natural Language Understanding (NLU)
- Natural Language Generation (NLG)



Natural Language Understanding (NLU)

Natural Language Understanding, is a subset of Natural Language Processing (NLP) that focuses on the comprehension of human language by machines.

NLU specifically deals with the ability of a system to comprehend the meaning and context of natural language input.

Key components of NLU include:

- Intent Recognition
- Entity Recognition
- Semantic Analysis
- Context Understanding
- Language Inference

Natural Language Generation (NLG)

Natural Language Generation, is a branch of natural language processing (NLP) that focuses on the automatic generation of human-readable and coherent natural language text by machines.

In NLG, systems take structured data or information as input and transform it into meaningful text in a way that mimics human language.

Examples of NLG applications include:

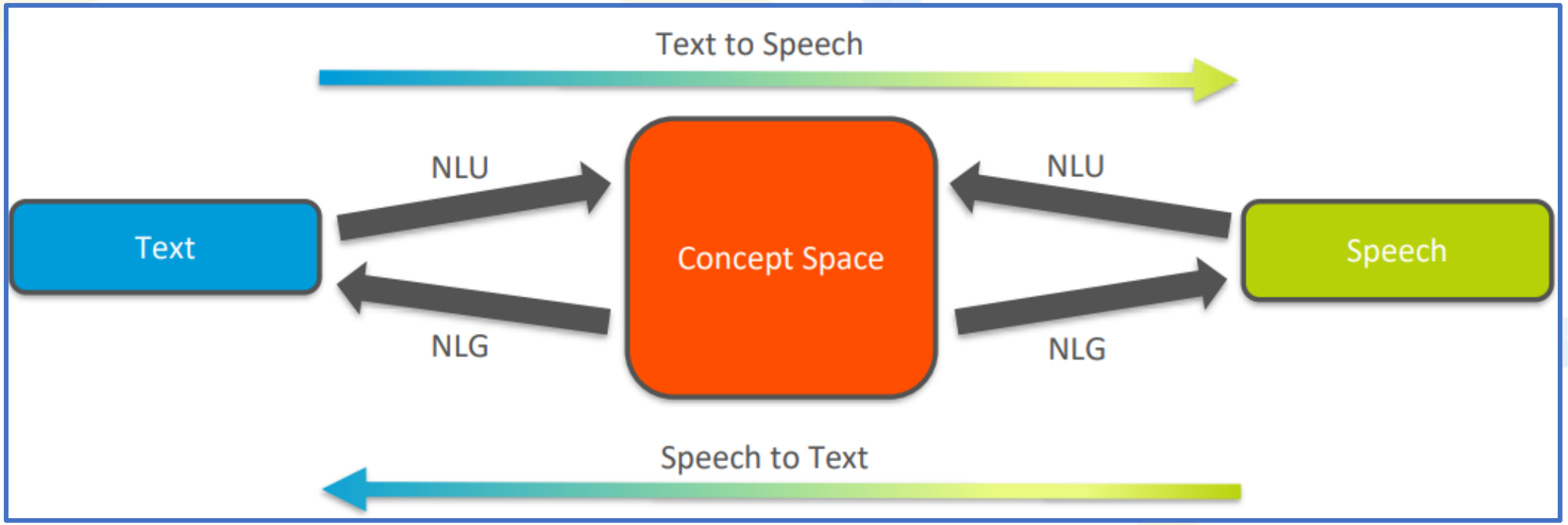
- Automated Reporting
- Chatbots and Virtual Assistants
- Content Creation
- Personalized Messaging

Natural Language Processing

Natural Language can refer to **Text or Speech**

Goal of both is the same:

Translate raw data (text or speech) into underlying concepts (NLU) then possibly into the other form (NLG)



History of NLP

NLP has been through (at least) 3 major eras:

1. 1950s-1980s: Linguistics Methods and Handwritten Rules
2. 1980s-Now: Corpus/Statistical Methods
3. Now-???: Deep Learning

History of NLP

1950s - 1980s: Linguistics/Rule Systems

NLP systems focus on:

- Linguistics: Grammar rules, sentence structure parsing, etc
- Handwritten Rules: Huge sets of logical (if/else) statements
- Ontologies: Manually created (domain-specific!) knowledge bases to augment rules above

Problems:

- Too complex to maintain
- Can't scale!
- Can't generalize!

History of NLP

1980s - Now: Corpus/Statistical Methods

NLP starts using Machine Learning methods

Use statistical learning over huge datasets of unstructured text

- Corpus: Collection of text documents
- e.g. Supervised Learning: Machine Translation
- e.g. Unsupervised Learning: Deriving Word "Meanings" (vectors)

History of NLP

Now - ???: Deep Learning

Deep Learning made its name with Images first

2012: Deep Learning has major NLP breakthroughs

- Researchers use a neural network to win the Large Scale Visual Recognition Challenge (LSVRC)
- This state of the art approach beat other ML approaches with half their error rate (26% vs 16%)

Very useful for unified processing of Language + Images

Important NLP Terms

Phonemes: the smallest sound units in a language

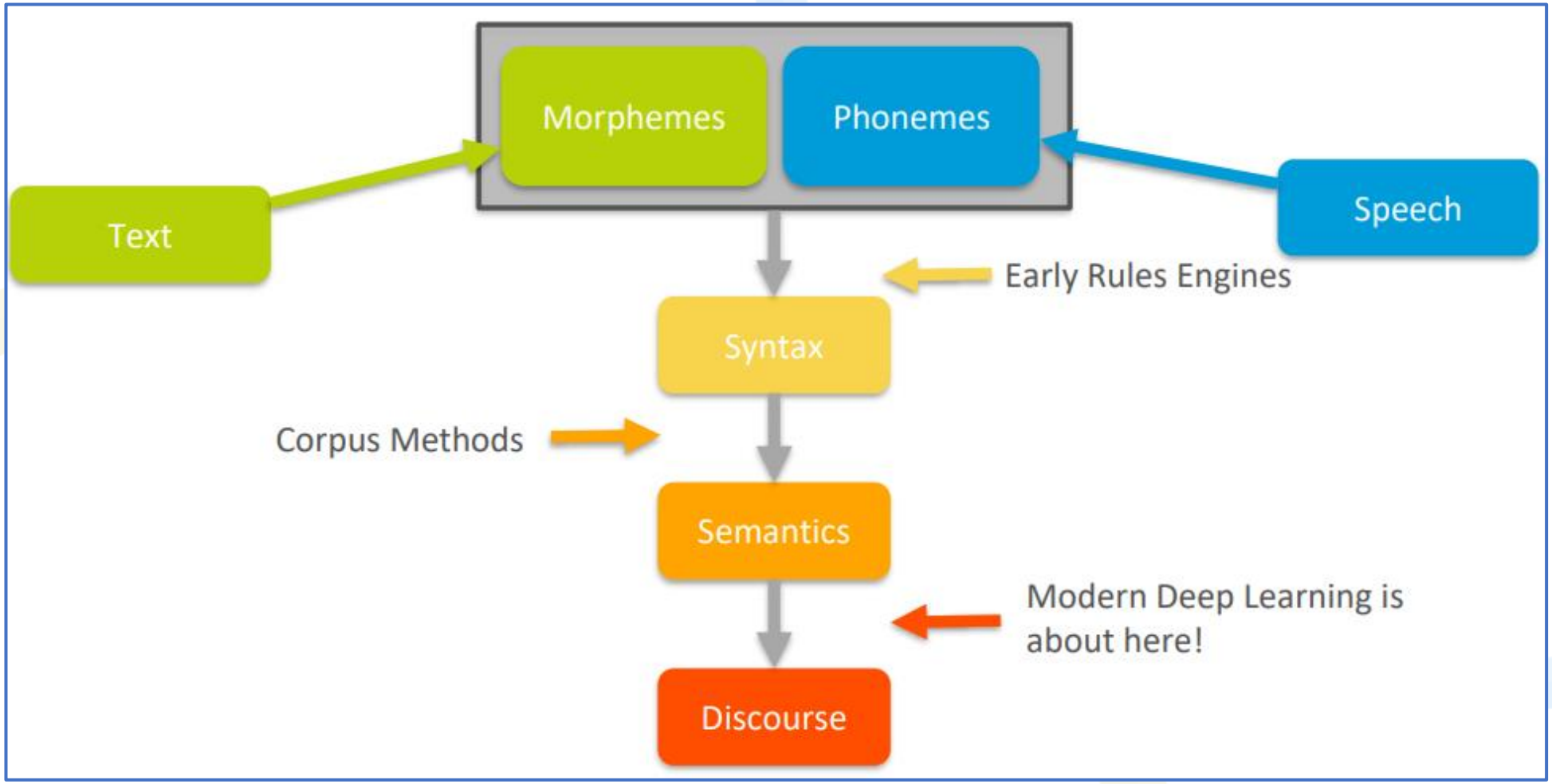
Morphemes: the smallest units of meaning in a language

Syntax: how words and sentences are constructed from these two building blocks

Semantics: the meaning of those words and sentences

Discourse: semantics in context. Conversation, persuasive writing, etc.

Levels of NLP



NLU Applications

ML on Text (Classification, Regression, Clustering)

Document Recommendation

Language Identification

Natural Language Search

Sentiment Analysis

Text Summarization

Extracting Word/Document Meaning (vectors)

Relationship Extraction

Topic Modeling

NLG Applications

Image Captioning

Text Summarization

Machine Translation

Question Answering/Chatbots

LUIS - Language Understanding Intelligent Solution

- Note: NLU is almost a prerequisite for NLG

Tokenization

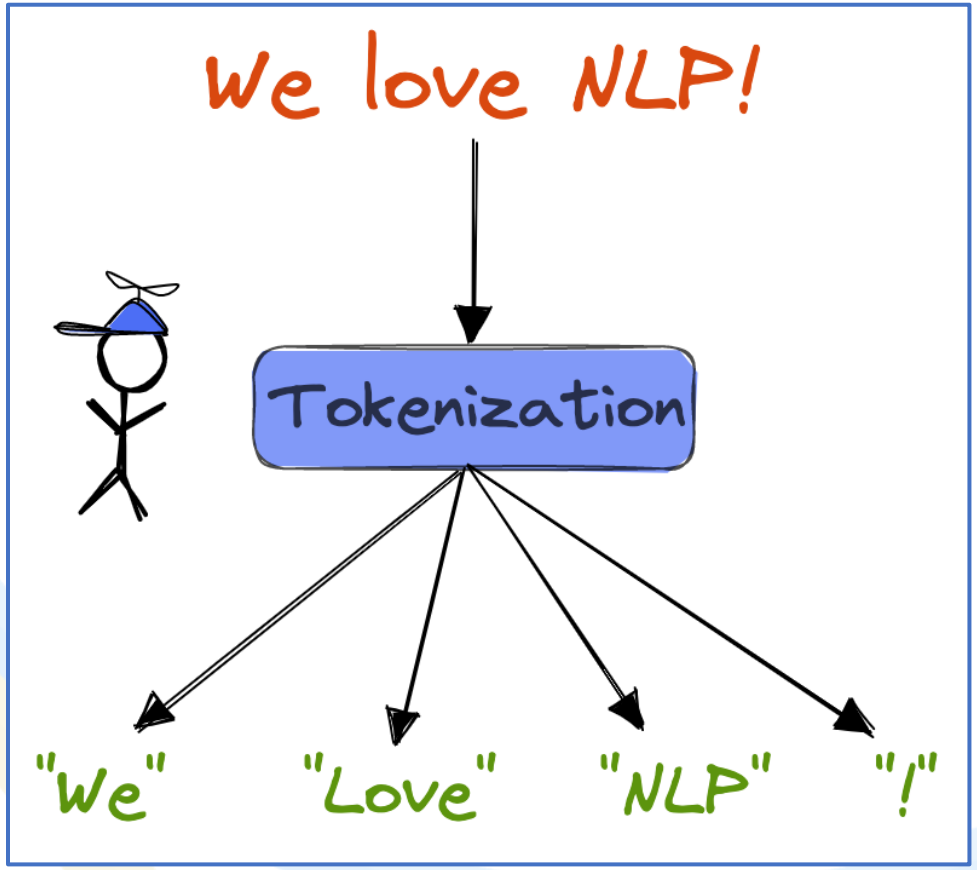
Tokenization

Tokenization is a fundamental preprocessing step in Natural Language Processing (NLP), and it involves breaking down a text into smaller units called tokens.

These tokens can be words, subwords, or even characters, depending on the granularity chosen.

Major Types:

- Character Tokenization
- Word Tokenization
- Sentence Tokenization



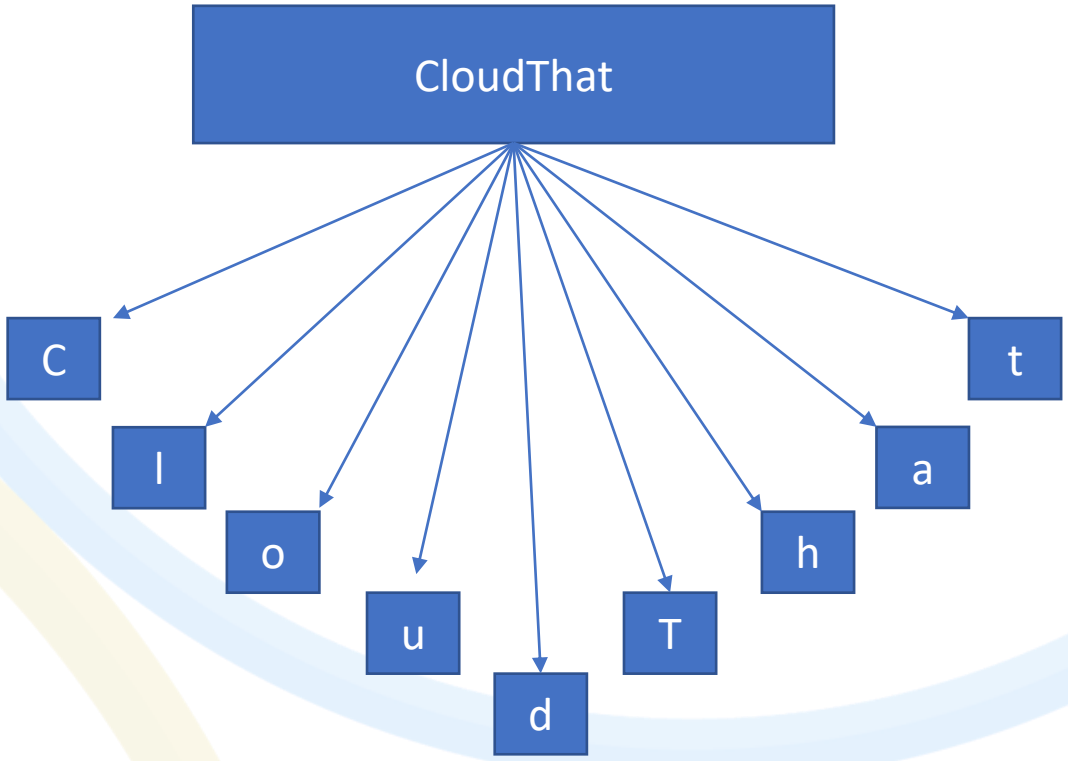
Character Tokenization

Character tokenization involves breaking down a text into individual characters.

This type of tokenization is more granular compared to word or sentence tokenization.

Each character in the text becomes a separate token.

Below is a simple example in Python using the NLTK library to demonstrate character tokenization.



Character Tokenization

cloudthat

Character Tokenization

```python
import nltk

def character_tokenize(text):
    # Using NLTK's built-in character tokenizer
    tokens = list(text)
    return tokens

# Example text
input_text = "Hi, I'm Kiran Dambal"

# Tokenize the text into characters
character_tokens = character_tokenize(input_text)

# Display the result
print(f"Original Text: {input_text}")
print(f"Character Tokens: {character_tokens}")
```

```
Original Text: Hi, I'm Kiran Dambal
Character Tokens: ['H', 'i', ',', ' ', 'I', "'", 'm', ' ', 'K', 'i', 'r', 'a', 'n', ' ', 'D', 'a', 'm', 'b', 'a', 'l']
```

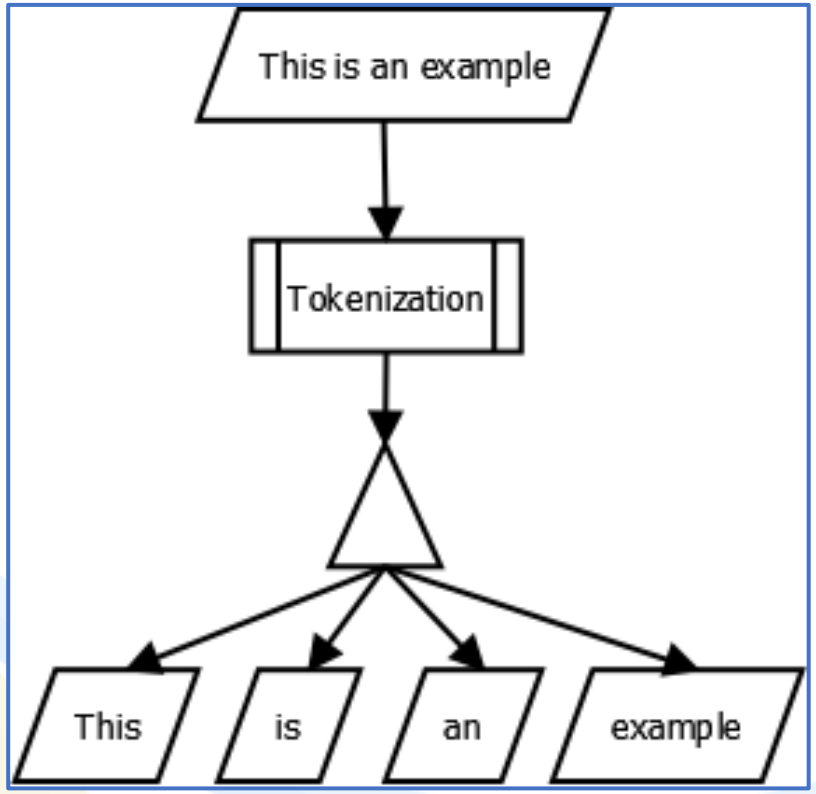
Word Tokenization

Word tokenization is the process of breaking a text into individual words or tokens.

This is a fundamental step in natural language processing (NLP) and is essential for various language-related tasks.

word tokenization is done using Python and the Natural Language Toolkit (NLTK) library.

The Punkt tokenizer is a pre-trained unsupervised machine learning model included in the Natural Language Toolkit (NLTK) library for tokenizing text into sentences or words.



Word Tokenization

```
import nltk
from nltk.tokenize import word_tokenize

nltk.download('punkt')

def word_tokenization(text):
    # Tokenize the text into words
    tokens = word_tokenize(text)
    return tokens

text = "We are AIML team of CloudThat"
tokens = word_tokenization(text)

print("Original Text:")
print(text)
print("\nWord Tokens:")
print(tokens)
```

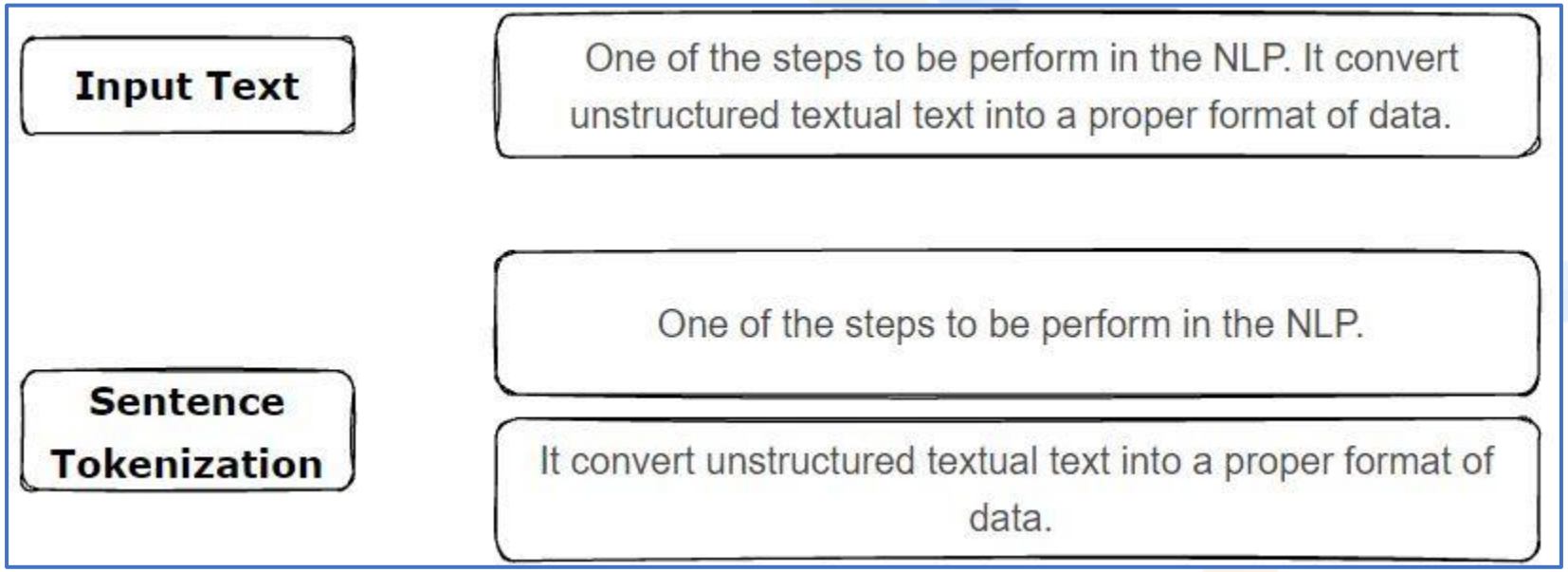
```
Original Text:
We are AIML team of CloudThat

Word Tokens:
['We', 'are', 'AIML', 'team', 'of', 'CloudThat']
```

Sentence Tokenization

Sentence tokenization is the process of breaking down a text into individual sentences.

This is a crucial step in natural language processing (NLP) and is often the first step in text processing pipelines.

Many NLP libraries, such as NLTK (Natural Language Toolkit), spaCy, and others, provide built-in functions for sentence tokenization.



Sentence Tokenization

```
# Import the necessary module from NLTK
from nltk.tokenize import sent_tokenize

# Sample text
text = "Hi. we are CloudThat. Training and Consulting Company"

# Tokenize the text into sentences
sentences = sent_tokenize(text)

# Print the result
for sentence in sentences:
    print(sentence)
```

```
Hi.
we are CloudThat.
Training and Consulting Company
```

cloudthat

Problem Statement - 1

Write a python code to perform character level, word level and sentence level Tokenization, read a txt document as a input

Tokenization

```python
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize

nltk.download('punkt')

txt_file_path = '/content/sample.txt'

with open(txt_file_path, 'r', encoding='utf-8') as file:
    text_data = file.read()

# Function for character-level tokenization
def char_tokenize(text):
    return list(text)

# Function for word-level tokenization
def word_tokenize_custom(text):
    return word_tokenize(text)

# Function for sentence-level tokenization
def sent_tokenize_custom(text):
    return sent_tokenize(text)
```

Tokenization

```
# Apply tokenization
char_tokens = char_tokenize(text_data)
word_tokens = word_tokenize_custom(text_data)
sent_tokens = sent_tokenize_custom(text_data)

print("\nCharacter-level Tokens:")
print(char_tokens)

print("\nWord-level Tokens:")
print(word_tokens)

print("\nSentence-level Tokens:")
print(sent_tokens)
```

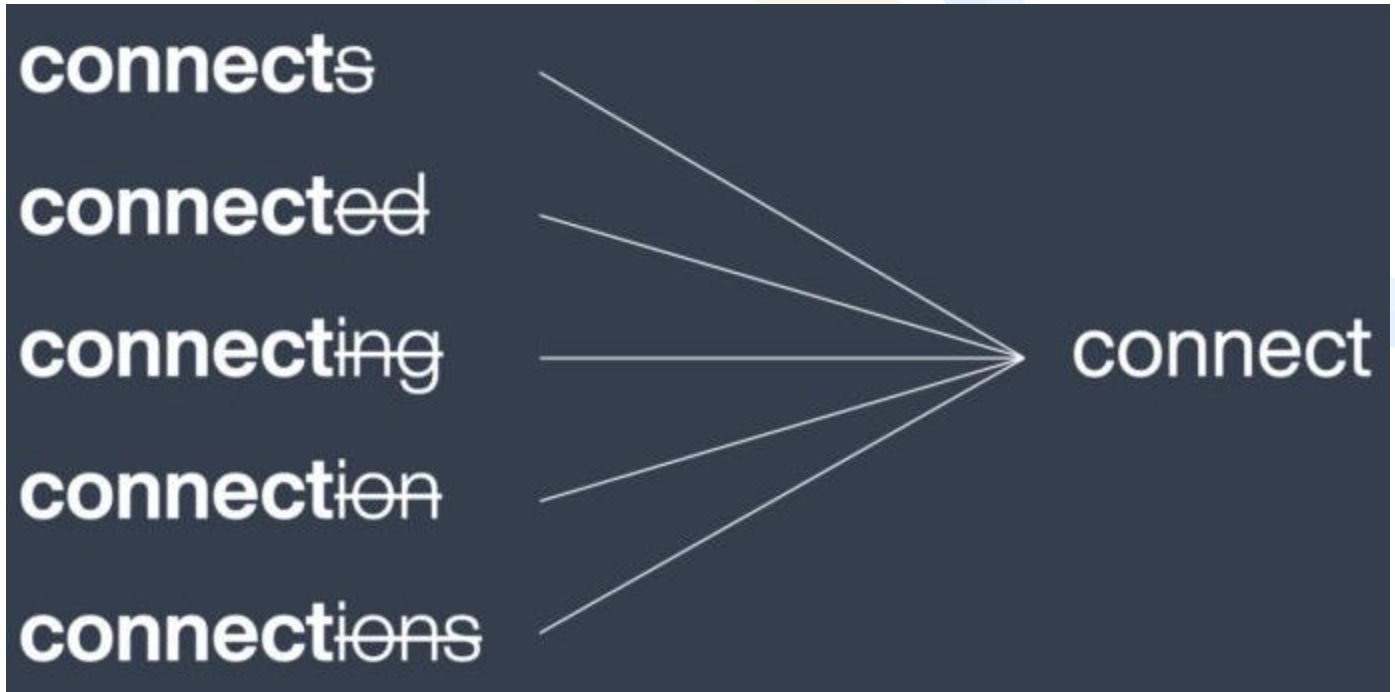
Text Lemmatization and Stemming

Stemming

Stemming is the process of removing prefixes or suffixes from words to obtain their root form, known as the "stem."

The goal of stemming is to reduce words to a common base form, even if the resulting stem is not a valid word.

Example:



Stemming

```
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
nltk.download('punkt')

sentence = "Stemming is a technique used in natural language processing."

# Tokenize the sentence into words
words = word_tokenize(sentence)

# Initialize the Porter Stemmer
porter_stemmer = PorterStemmer()

# Perform stemming on each word
stemmed_words = [porter_stemmer.stem(word) for word in words]

# Display the original words and their stems
print("Original Words:", words)
print("Stemmed Words:", stemmed_words)
```

Stemming

```
Original Words: ['Stemming', 'is', 'a', 'technique', 'used', 'in', 'natural', 'language', 'processing', '.']
Stemmed Words: ['stem', 'is', 'a', 'techniqu', 'use', 'in', 'natur', 'languag', 'process', '.']
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Lemmatization

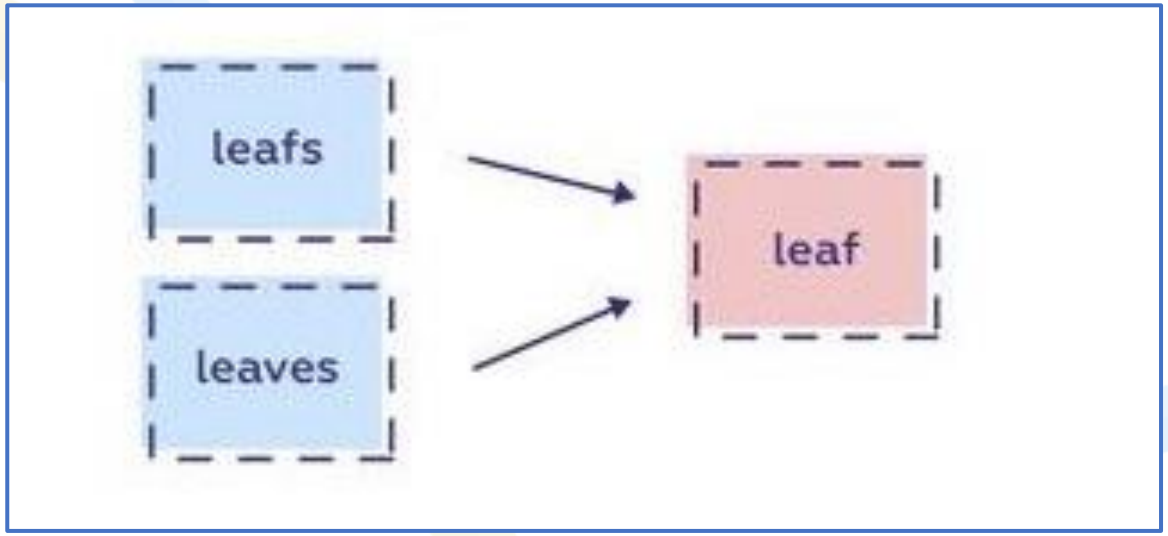
Lemmatization, on the other hand, is a more sophisticated process that involves reducing words to their base or dictionary form, known as the "lemma."

The lemmatized form is a valid word and represents the canonical form of a word.

Example:

Original: better, best

Lemma: good



Lemmatization

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

# Example sentence
sentence = "Lemmatization is helping me in understanding NLP."

# Tokenize the sentence into words
words = word_tokenize(sentence)

# Initialize the WordNet Lemmatizer
lemmatizer = WordNetLemmatizer()

# Perform lemmatization on each word
lemmatized_words = [lemmatizer.lemmatize(word) for word in words]

# Display the original words and their lemmatized forms
print("Original Words:", words)
print("Lemmatized Words:", lemmatized_words)
```

Lemmatization v/s Stemming

cloudthat

Stemming

achieve -> achiev
achieving -> achiev

- Can reduce words to a stem that is not an existing word
- Operates on a single word without knowledge of the context
- Simpler and faster

Lemmatization

achieve -> achieve
achieving -> achieve

- Reduces inflected words to their lemma, which is always an existing word
- Can leverage context to find the correct lemma of a word
- More accurate but slower