

Why we are installing spacy??

→ it is used for sentiment and other analysis and much faster than nltk as well here

→ we can say that it is much more faster here.

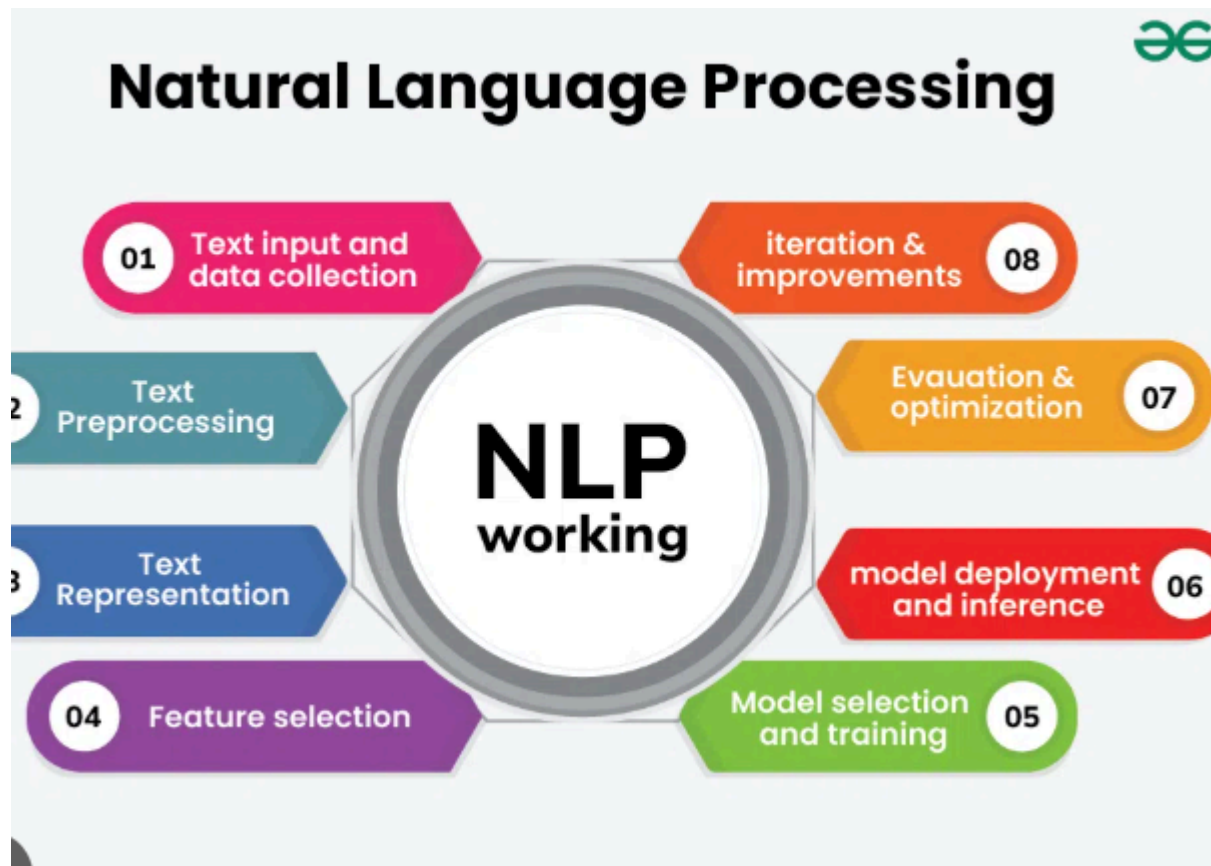
What is pyPDF2 is for ??

→ it is for accessing the pdf and all here

→ we can access write and read file easily here

→ and use the file accessing and all here easily..

How the nlp is basically works here



→ here you can see by this way we can say how nlp is basically works here

→ nlp using pipeline object to basically tokenized , learn and then establishment for all itis actually using this here...

→how the tokenization is actually having

What is Tokenization?

A process that involves transforming sensitive data into nonsensitive data.

Data that is tokenized becomes undecipherable and irreversible. No mathematical relationships exist between tokens and the original data.



→ so here is the process for the tokenization and how it is actually working that i have explained here in a simpler definition..

→ so everything word or anything that is useful to understand actually the word and its actual meaning like how and what is actually the word is for that is called the tokenization actually and there relationship to understand with other words.

What is stemmer ??

→ Now you think what is stemmer

→ A **stemmer** is a tool or algorithm in Natural Language Processing (NLP) that **reduces a word to its base or root form** (called the *stem*) by removing common suffixes or prefixes.

→ so here is the basics of the stemmers like how it is and what is this for

→  **Example:**

Original Word	Stemmed Word
---------------	--------------

running	run
---------	-----

runs	run
------	-----

easily	easili
--------	--------

studies	studi
---------	-------

Concern

Description

- ♦ **Over-stemming** Might reduce too much (**university** → **univers**)
- ♦ **Loss of meaning** Stems may not be valid words (**easili** isn't a real word)

- ♦ **Ambiguity** Different words may reduce to the same stem (**general**, **generate** → **gener**)

Lemmatization means reducing a word to its **base or dictionary form** (called a **lemma**) using the **word's meaning and grammar**.

👉 Simple Example:

- "running" → "run"
- "better" → "good"

✅ It uses context (like part of speech), unlike stemming which just chops the word.

So, **lemmatization** = **smart simplification** based on meaning.

Ask ChatGPT

We do **lemmatization** to make **text easier for machines to understand**. Here's the simple reason:

🧠 Why Lemmatization?

When we process text in **NLP (Natural Language Processing)**, we want to **group similar words** together. Different forms of a word (like "run", "running", "ran") mean the **same basic thing**. But a computer sees them as **different** unless we normalize them.

📌 Real-World Example:

Without lemmatization:

"He **runs** fast. He **ran** yesterday."

The computer sees: **runs** ≠ **ran**

✅ After lemmatization: both become "**run**" → easier for machines to **analyze, search, summarize, or train models**.

💡 Benefits:

- Reduces word **variations**

- Makes **search engines**, **chatbots**, and **AI models** smarter
 - Helps with **text classification**, **sentiment analysis**, etc.
-

Lemmatization vs Stemming:

- **Stemming** is faster but less accurate.
 - **Lemmatization** is slower but understands grammar and meaning.
-

In short:


 **Lemmatization cleans up language** so machines can better understand what we really mean.

Now who understand the grammar then ??

Great question. When we want to **generate** text or understand **grammar** deeply (like forming correct sentences), we need more than just lemmatization or stemming.

Here's how it works in simple terms:

Who understands grammar behind the sentence?

 **Language models** and **NLP pipelines** do.

Tools that understand grammar:

1. **spaCy**
 2. **Transformers** (like BERT, GPT, etc.)
 3. **Part-of-Speech (POS) tagging**
 4. **Dependency Parsing**
-

How grammar is understood:

1. **POS tagging** → identifies if a word is a noun, verb, adjective, etc.
 2. **Dependency parsing** → finds how words relate to each other in a sentence.
 3. **Syntax trees** → models build a structure of the sentence (subject, verb, object).
 4. **Named Entity Recognition (NER)** → understands what is a name, place, money, etc.
-

Example:

Sentence:

"Apple is going to build a U.K. factory for \$6 million."

spaCy or GPT understands:

- **"Apple"** → Subject (NOUN, ORG)
- **"is going"** → Verb (tense: present continuous)
- **"to build"** → Infinitive verb (future intent)
- **"\$6 million"** → MONEY (amount)

So a model like **GPT** can then say:

"Apple plans to construct a facility in the UK costing six million dollars."

✅ Grammar understood. ✅ New text generated.

In summary:

Lemmatization **helps clean the words**,
but models like GPT or spaCy's parser **understand the grammar and meaning** to generate or process text correctly.

Let me know if you want a demo of how grammar is parsed in your sentence.

Now what is the mean of the dependency parsing then ??

Dependency parsing means **finding how words are connected in a sentence**, like who is doing what to whom.

Simple definition:

Dependency parsing finds the **grammatical structure** of a sentence by identifying the **head word** and how other words are **dependent** on it.


Example:

Sentence:

"Raj eats mango."

Dependency parse tells us:

- **"eats"** is the **main verb** (root)
- **"Raj"** is the **subject** of "eats"
- **"mango"** is the **object** of "eats"

 It shows:

scss

Copy code

eats ← Raj (subject)

eats → mango (object)

Why is it useful?

- It helps machines **understand sentence meaning**
 - Used in **chatbots, text summarization, question answering**, etc.
-

Think of it like:

A sentence is a **puzzle**, and dependency parsing helps the machine **put the grammar pieces together correctly**.

Let me know if you want to visualize it with [spaCy](#)!

→ now dependency parsing is useful for parsing the grammar and correlate like how the grammar is actually use full for here