```
!pip install spacy
!python -m spacy download en_core_web_sm
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.11/dist-packages (3.8.7)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.13)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.10)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (8.3.6)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.16.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (4.67.1)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.32.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.11.7)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (25.0)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.5.0)
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>=3.2.0->spacy) (1.3.
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.5.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2025
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.3.0)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (0.
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (8.2.1)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (13.9.4)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy) (7
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (3.0.2)
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data>=1.2->langcodes<4.0.0,>
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0.0,>=0.3.0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0.0,>=0.3
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.1.0->sp
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-any.whl (12
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 92.3 MB/s eta 0:00:00
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

```
import spacy
nlp = spacy.load("en_core_web_sm")
```

Explanation: You first install spaCy and download a small English language model (en_core_web_sm).

import spacy brings the library into your script.

nlp = spacy.load("en_core_web_sm") loads the pre-trained model, which is required for all subsequent analysis. This setup step is necessary for any spaCy-based application.

Tokenization

```
text = "spaCy is an advanced NLP library."
doc = nlp(text)
for token in doc:
    print(token.text)
```

```
spaCy
is
an
advanced
```

```
NLP
library
.
```

Explanation: Tokenization is splitting a text into basic units called tokens (words, punctuation, etc.).

doc = nlp(text) processes the input string, producing a spaCy Doc object.

Iterating over doc yields each token as an object; token.text gives the original text for each token.

This step is foundational, enabling detailed analysis of text structure and meaning.

Sentence Segmentation

```
text = "Natural Language Processing enables computers to understand human language. It's a fast-growing field!"
doc = nlp(text)
for sent in doc.sents:
    print(sent.text)
```

```
Natural Language Processing enables computers to understand human language.
It's a fast-growing field!
```

Explanation: Sentence segmentation divides text into sentences.

Each sent in doc.sents is a sentence object.

Accurate sentence splitting is necessary before more advanced tasks (e.g., parsing, information extraction).

Part-of-Speech (POS) Tagging

```
text = "spaCy processes text efficiently."
doc = nlp(text)
for token in doc:
    print(f"{token.text} - {token.pos_}")
```

```
spaCy - X
processes - NOUN
text - NOUN
efficiently - ADV
. - PUNCT
```

Explanation: POS tagging labels each word with its grammatical category (noun, verb, etc.).

token.pos_ provides the POS tag (e.g., 'NOUN', 'VERB').

This process is crucial for understanding syntax and structure, supporting tasks like parsing and information extraction

Lemmatization

```
text = "The leaves are falling."
doc = nlp(text)
for token in doc:
    print(f"{token.text} - Lemma: {token.lemma_}")
```

```
The - Lemma: the
leaves - Lemma: leave
are - Lemma: be
falling - Lemma: fall
. - Lemma: .
```

Explanation: Lemmatization reduces each word to its dictionary (base) form.

For example, "leaves" → "leaf", "falling" → "fall".

This helps normalize vocabulary, supporting search and analysis tasks.

Named Entity Recognition (NER)

```
text = "Apple is looking at buying U.K. startup for $1 billion."
doc = nlp(text)
for ent in doc.ents:
    print(f"Entity: {ent.text}, Label: {ent.label_}")
```

```
Entity: Apple, Label: ORG
Entity: U.K., Label: GPE
Entity: $1 billion, Label: MONEY
```

Explanation: NER identifies real-world entities like names, places, and monetary amounts in text.

Each detected entity (ent) has a label (e.g., 'ORG', 'GPE', 'MONEY').

Recognizing entities enables applications like information extraction, search, and question answering.

Stop Word Detection

```
text = "This is a simple test."
doc = nlp(text)
for token in doc:
    print(f"{token.text} - Stopword: {token.is_stop}")
```

```
This - Stopword: True
is - Stopword: True
a - Stopword: True
simple - Stopword: False
test - Stopword: False
. - Stopword: False
```

Explanation: Stop words are common words (like "is", "a", "the") that usually carry little semantic meaning.

token.is_stop is True for stop words.

Removing stop words simplifies text and focuses analysis on keywords.

Dependency Parsing

```
text = "spaCy lets you analyze linguistic structure easily."
doc = nlp(text)
for token in doc:
    print(f"{token.text} <--{token.dep_}-- {token.head.text}")
```

```
spaCy <--meta-- lets
lets <--ROOT-- lets
you <--nsubj-- analyze
analyze <--ccomp-- lets
linguistic <--amod-- structure
structure <--dobj-- analyze
easily <--advmod-- analyze
. <--punct-- lets
```

Explanation: Dependency parsing analyzes sentence structure by describing how words depend on each other.

token.dep_ is the type of grammatical relationship (subject, object, etc.).

token.head.text is the word this token depends on.

Understanding dependencies is key for extracting who does what to whom, supporting tasks like translation and question-answering

Combined Workflow Example

```
text = "Barack Obama was born in Hawaii. He was elected president in 2008."
doc = nlp(text)

# Sentences
for sent in doc.sents:
    print("Sentence:", sent.text)
    for token in sent:
        print(f"  {token.text}: POS={token.pos_}, Lemma={token.lemma_}, Stopword={token.is_stop}")
    for ent in sent.ents:
        print(f"  Entity: {ent.text}, Label: {ent.label_}")
```

```
Sentence: Barack Obama was born in Hawaii.
  Barack: POS=PROPN, Lemma=Barack, Stopword=False
  Obama: POS=PROPN, Lemma=Obama, Stopword=False
  was: POS=AUX, Lemma=be, Stopword=True
  born: POS=VERB, Lemma=bear, Stopword=False
  in: POS=ADP, Lemma=in, Stopword=True
  Hawaii: POS=PROPN, Lemma=Hawaii, Stopword=False
  .: POS=PUNCT, Lemma=., Stopword=False
  Entity: Barack Obama, Label: PERSON
  Entity: Hawaii, Label: GPE
Sentence: He was elected president in 2008.
  He: POS=PRON, Lemma=he, Stopword=True
  was: POS=AUX, Lemma=be, Stopword=True
  elected: POS=VERB, Lemma=elect, Stopword=False
  president: POS=NOUN, Lemma=president, Stopword=False
  in: POS=ADP, Lemma=in, Stopword=True
  2008: POS=NUM, Lemma=2008, Stopword=False
  .: POS=PUNCT, Lemma=., Stopword=False
  Entity: 2008, Label: DATE
```

Explanation: This program combines several spaCy features:

Sentence segmentation: Splits the text into individual sentences.

Token-based analysis: Prints POS, lemma, and stop word status for each word.

NER per sentence: Shows entities found in each sentence.

This type of analysis is foundational for many advanced NLP applications as described in the reference text

```
Start coding or generate with AI.
```