Natural Language Processing using Python Programming

Notebook 01.2: Installation and Environment Setup

Python 3.8+ NLTK Latest SpaCy Latest License MIT

Part of the comprehensive learning series: Natural Language Processing using Python Programming

Learning Objectives:

- Install and configure NLTK library for basic NLP tasks
- Set up SpaCy with language models for advanced processing
- Verify installations with practical examples
- Prepare the development environment for upcoming NLP projects

---

- This notebook is our first hands-on step.

- We will cover the practical aspects of setting up the two most important Python libraries for this course: **NLTK** (Natural Language Toolkit) and **SpaCy**.

- We will install necessary language data and run simple code to verify everything is working correctly.

1. Prerequisites Check (Non-Executable)

- Before running the code cells below, please ensure you have completed the following steps from the `README.md`:

    1. **Cloned** the repository and navigated to the directory.

    2. **Activated** your Python virtual environment (`nlp_env`).

    3. **Installed** all dependencies using `pip install -r requirements.txt`.

2. Setting Up NLTK (Natural Language Toolkit)

- **NLTK** is the oldest and most commonly used library for academic research, education, and foundational NLP concepts.

- It includes access to hundreds of corpora and lexical resources.

- While the `pip install` covers the library itself, NLTK requires manually downloading specific datasets, tokenizers, and resources.

- We will download the essential ones here.

In [1]:
```python
import nltk

print("Starting NLTK downloads...")

# 1. 'punkt': Required for word and sentence tokenization
nltk.download('punkt', quiet=True)

# 2. 'stopwords': A list of common words to ignore in analysis
nltk.download('stopwords', quiet=True)

# 3. 'wordnet': Used for advanced analysis like Lemmatization
nltk.download('wordnet', quiet=True)

# 4. 'averaged_perceptron_tagger': Required for Part-of-Speech (POS) tagging
nltk.download('averaged_perceptron_tagger', quiet=True)

print("\nNLTK essential resources downloaded successfully!")
```

```
Starting NLTK downloads...

NLTK essential resources downloaded successfully!
```

NLTK Verification: Our First NLP Program (Tokenization)

- Let's confirm NLTK is working by performing the fundamental task of **Word Tokenization**.

In [2]:
```python
from nltk.tokenize import word_tokenize

text_sample = "NLP is fascinating, and I'm ready to learn more!"

# Use NLTK's word_tokenize function
tokens = word_tokenize(text_sample)

print(f"Original Text: {text_sample}")
print(f"\nTokens Output: {tokens}")
```

```
Original Text: NLP is fascinating, and I'm ready to learn more!

Tokens Output: ['NLP', 'is', 'fascinating', ',', 'and', 'I', "'m", 'ready', 'to',
'learn', 'more', '!']
```

3. Setting Up SpaCy

- **SpaCy** is an industrial-strength library designed for efficiency, speed, and production use.

- Unlike NLTK, which uses separate packages for different tasks, SpaCy uses a unified **language model** that encapsulates all components (tokenization, POS tagging, NER, etc.).

- We need to download the English language model, typically the small one (`en_core_web_sm`) for quick testing and basic tasks.

In [4]:
```
import spacy

# SpaCy models are downloaded using the command line (or shell commands in the not
# The '!' prefix runs the command in the shell
print("Downloading SpaCy English model 'en_core_web_sm'...")

!python -m spacy download en_core_web_sm

print("\nSpaCy model download complete!")
```

```
Downloading SpaCy English model 'en_core_web_sm'...
Collecting en-core-web-sm==3.8.0
SpaCy model download complete!

  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_w
eb_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-any.whl (12.8 MB)
     ---------------------------------------- 0.0/12.8 MB ? eta -:--:--
     ---- ----------------------------------- 1.3/12.8 MB 7.3 MB/s eta 0:00:02
     -------- ------------------------------- 2.6/12.8 MB 7.2 MB/s eta 0:00:02
     -------------- ------------------------- 4.7/12.8 MB 7.9 MB/s eta 0:00:02
     -------------------- ------------------- 6.6/12.8 MB 8.1 MB/s eta 0:00:01
     ------------------------ --------------- 7.9/12.8 MB 7.9 MB/s eta 0:00:01
     --------------------------- ------------ 8.7/12.8 MB 7.1 MB/s eta 0:00:01
     ---------------------------- ----------- 9.2/12.8 MB 6.6 MB/s eta 0:00:01
     ------------------------------ --------- 9.7/12.8 MB 6.2 MB/s eta 0:00:01
     -------------------------------- ------- 10.5/12.8 MB 5.7 MB/s eta 0:00:01
     ------------------------------------ --- 11.5/12.8 MB 5.6 MB/s eta 0:00:01
     ---------------------------------------  12.6/12.8 MB 5.6 MB/s eta 0:00:01
     ---------------------------------------  12.6/12.8 MB 5.6 MB/s eta 0:00:01
     ---------------------------------------- 12.8/12.8 MB 4.7 MB/s  0:00:02
Installing collected packages: en-core-web-sm
Successfully installed en-core-web-sm-3.8.0
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

SpaCy Verification: Our First NLP Program (Pipeline)

- SpaCy processes text by passing it through the loaded model, creating a **Doc** object.
- This object holds all the processed information, including tokens, POS tags, and named entities.

In [7]:
```
# 1. Load the model
nlp = spacy.load("en_core_web_sm")

# 2. Process the text to create a Doc object
doc = nlp("Apple is looking at buying U.K. startup for $1 billion.")

# 3. Iterate over the Doc object to see tokens and their properties
print("Token    | Lemma    |  POS Tag  | Named Entity Type")
print("--------|---------|-----------|------------------")
for token in doc:
    # token.text: The original word/punctuation
    # token.lemma_: The base form of the word (lemma)
    # token.pos_: The Part-of-Speech tag
    print(f"{token.text:<5}   | {token.lemma_:<5}   | {token.pos_:<7}   | {token.e
Token    | Lemma    |  POS Tag   | Named Entity Type
--------|---------|-----------|------------------
Apple    | Apple    | PROPN      | ORG
is       | be       | AUX        |
looking    | look     | VERB       |
at       | at       | ADP        |
buying    | buy      | VERB       |
U.K.     | U.K.     | PROPN      | GPE
startup    | startup    | VERB       |
for      | for      | ADP        |
$        | $        | SYM        | MONEY
1        | 1        | NUM        | MONEY
billion    | billion    | NUM        | MONEY
.        | .        | PUNCT      |
```

4. Summary and Next Steps

- If both verification steps ran without error, your environment is successfully set up!
- You now have the necessary tools to begin processing language.
- In **Chapter 2**, we will take a deep dive into **Text Preprocessing**, learning how to apply the techniques we verified here (tokenization, lemmatization) in detail, along with other essential cleaning steps.

Key Takeaways

- **Environment Setup:** We successfully installed and configured NLTK and SpaCy libraries for NLP development.
- **Library Verification:** Both NLTK and SpaCy are working correctly with proper language models and data downloads.
- **Development Ready:** Our environment is now prepared for hands-on NLP implementation and experimentation.

Next Notebook Preview

- With our environment set up, we're ready to dive into **practical text preprocessing techniques**.
- The next notebook will cover **tokenization, text cleaning, and normalization** - the essential first steps in any NLP pipeline.

About This Project

This notebook is part of the **Natural Language Processing using Python Programming for Beginners** repository - a comprehensive, beginner-friendly guide for mastering NLP using Python, NLTK, and SpaCy.

Repository: `NLP`

Author

Prakash Ukhalkar
GitHub prakash-ukhalkar

Built with ❤️ for the Python community