Web Scraping and NLP Preprocessing with Python

Andrés Mendez Vázquez

August 12, 2025

What Will You Learn Today?

- Fundamentals of Web Scraping with Python
- Tools like Selenium, BeautifulSoup, and Requests
- Collecting and preparing text for Natural Language Processing (NLP)
- Cleaning, spelling correction, and basic entity recognition
- Hands-on examples in JupyterLab

What is Web Scraping?

- Extracting structured data from web pages
- Use cases: news analysis, price tracking, social media mining
- Legal and ethical considerations (robots.txt, usage limits)

Basic Techniques and Tools

- **requests**: fetch HTML content (great for static pages)
- **BeautifulSoup**: parse and navigate HTML
- **Selenium**: automate interaction with dynamic pages (JS-heavy, login flows)
- When to use each tool

Practical Example: Scraping Headlines

- Target: news site or Wikipedia
- Code example using `requests + BeautifulSoup`
- Demo in JupyterLab

`requests` – Fetching Web Content

The `requests` library is one of the most straightforward ways to retrieve data from the web using HTTP. It allows us to send GET and POST requests and handle the responses easily.

- A simple and powerful library to send HTTP requests in Python.
- Used to download the HTML content of a web page.
- Ideal for static pages that do not require JavaScript rendering.

`BeautifulSoup` – Parsing HTML

After fetching raw HTML content, we need to extract relevant information. `BeautifulSoup` helps us parse the structure and navigate the DOM tree with ease.

- Parses HTML and XML documents.
- Allows navigation via tag names, classes, attributes.
- Useful for extracting text, links, headings, and more.

NLTK – Natural Language Toolkit

`NLTK` is a foundational library for NLP education and prototyping. It provides tools and datasets for linguistic analysis and preprocessing.

- Includes corpora, stopword lists, and basic NLP tools.
- Great for learning and quick experimentation.
- We'll use it to remove stopwords in our pipeline.

SpaCy – Modern NLP Toolkit

`SpaCy` is a fast and production-ready NLP library used for advanced text processing. It allows you to tokenize, lemmatize, and analyze text efficiently. More roboust than NLTK.

- Supports tokenization, POS tagging, lemmatization, NER.
- Highly efficient and accurate.
- We'll use it for cleaning and standardizing scraped text.

SymSpell – Fast Spelling Correction

When dealing with user generated or scraped text, spelling mistakes are common. `SymSpell` is a fast correction tool using edit distance and word frequency.

- Uses frequency based dictionaries for suggestions.
- Performs fast lookups using a hash-based algorithm.
- Helps fix common typos with high efficiency.

Regular Expressions – Pattern Matching

Regular expressions are a flexible way to detect text patterns such as emails, phone numbers, and dates. They're ideal for lightweight rule-based extraction.

- Detect patterns like emails, dates, and prices.
- Flexible and language-agnostic.
- Can be combined with other tools like FlashText.

FlashText – Efficient Keyword Search

Looking for predefined keywords in large texts can be slow with regex.
`FlashText` offers a high-performance alternative for keyword extraction.

- Extracts keywords faster than regular expressions.
- Scales well to long documents.

WordCloud – Text Visualization

`WordCloud` is a visual representation of word frequency in a corpus. It's a great tool to quickly understand which terms appear most in your data.

- Highlights the most frequent tokens.
- Visually engaging and easy to interpret.
- Helps summarize large amounts of text at a glance.

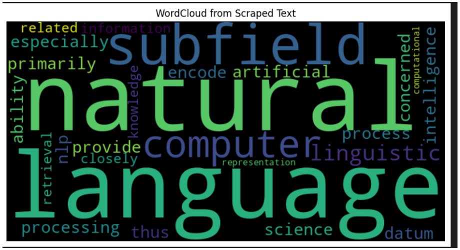


Figure: Output example of WordCloud

Web Scraping and NLP Pipeline

Web Page (HTML)

↓

Fetch with requests

↓

Parse with BeautifulSoup → Extract text

↓

Clean (SpaCy + NLTK)

↓

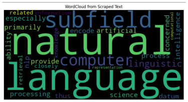
Spell correction (SymSpell)

↓

Entity Extraction (Regex + FlashText)

↓

Visualization (WordCloud)



Examples in Jupyter

- Step-by-step data extraction
- Text cleaning and analysis
- Simple visualizations (wordclouds, histograms)
- **Headline sentiment with NLTK VADER (positive/neutral/negative)**

Sentiment Analysis with NLTK VADER

- Lexicon-based sentiment for short English texts (headlines, tweets)
- Outputs `pos`, `neu`, `neg`, and `compound` ($[-1, 1]$)
- Simple thresholds: compound ≥ 0.05 positive, ≤ -0.05 negative, else neutral

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download("vader_lexicon")
sia = SentimentIntensityAnalyzer()

text = "Chip␣stocks␣rally␣after␣strong␣earnings"
scores = sia.polarity_scores(text) # {'neg':..., 'neu':..., 'pos':...,
     'compound':...}
```