```
pip install nltk spacy
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/d
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/py
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/pyth
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/pyt
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/pyth
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/l
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pytho
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/py
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/pyt
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.1
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/pyt
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/p
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/pyt
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packag
```

```
import nltk
import spacy

# Download NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# Download spaCy language model
spacy.cli.download('en_core_web_sm')

print("NLTK data and spaCy model downloaded successfully.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
```

```
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
✓ 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 th
'Restart kernel' or 'Restart runtime' option.
NLTK data and spaCy model downloaded successfully.
```

```
sample_essay = """The quick brown fox jumps over the lazy dog. This is a cla

print("Sample essay defined successfully.")
```

```
Sample essay defined successfully.
```

```
import nltk
import spacy

# Download NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt_tab') # Added to download the missing resource

# Download spaCy language model
spacy.cli.download('en_core_web_sm')

print("NLTK data and spaCy model downloaded successfully.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
✓ 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 th
'Restart kernel' or 'Restart runtime' option.
NLTK data and spaCy model downloaded successfully.
```

```
sample_essay = """The quick brown fox jumps over the lazy dog. This is a cla

from nltk.tokenize import sent_tokenize, word_tokenize

# Tokenize into sentences
nltk_sentences = sent_tokenize(sample_essay)

# Tokenize into words
```

```
nltk_words = word_tokenize(sample_essay)

print("NLTK Sentence Tokenization (first 3 sentences):")
for i, sentence in enumerate(nltk_sentences[:3]):
    print(f"Sentence {i+1}: {sentence}")

print("\nNLTK Word Tokenization (first 20 words):")
print(nltk_words[:20])
```

```
NLTK Sentence Tokenization (first 3 sentences):
Sentence 1: The quick brown fox jumps over the lazy dog.
Sentence 2: This is a classic sentence used to display all letters of the alp
Sentence 3: Natural Language Processing (NLP) involves understanding and mani

NLTK Word Tokenization (first 20 words):
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', '.',
```

```
from nltk import pos_tag

# Apply NLTK's POS tagger
nltk_pos_tags = pos_tag(nltk_words)

print("NLTK Part-of-Speech Tagging (first 20 tuples):")
print(nltk_pos_tags[:20])
```

```
NLTK Part-of-Speech Tagging (first 20 tuples):
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'V
```

```
import nltk
nltk.download('averaged_perceptron_tagger_eng')

print("NLTK averaged_perceptron_tagger_eng downloaded successfully.")
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
NLTK averaged_perceptron_tagger_eng downloaded successfully.
```

```
sample_essay = """The quick brown fox jumps over the lazy dog. This is a cla

from nltk.tokenize import sent_tokenize, word_tokenize
from nltk import pos_tag

# Tokenize into words (re-executing for kernel state consistency)
nltk_words = word_tokenize(sample_essay)

# Apply NLTK's POS tagger
nltk_pos_tags = pos_tag(nltk_words)

print("NLTK Part-of-Speech Tagging (first 20 tuples):")
print(nltk_pos_tags[:20])
```

```
NLTK Part-of-Speech Tagging (first 20 tuples):
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'V
```

```
import spacy

# Load the spaCy language model
nlp = spacy.load('en_core_web_sm')

# Process the sample essay text
spacy_doc = nlp(sample_essay)

print("spaCy Tokenization and POS Tagging (first 20 tokens):")
for i, token in enumerate(spacy_doc[:20]):
    print(f"Token {i+1}: {token.text} ({token.pos_})")
```

```
spaCy Tokenization and POS Tagging (first 20 tokens):
Token 1: The (DET)
Token 2: quick (ADJ)
Token 3: brown (ADJ)
Token 4: fox (NOUN)
Token 5: jumps (VERB)
Token 6: over (ADP)
Token 7: the (DET)
Token 8: lazy (ADJ)
Token 9: dog (NOUN)
Token 10: . (PUNCT)
Token 11: This (PRON)
Token 12: is (AUX)
Token 13: a (DET)
Token 14: classic (ADJ)
Token 15: sentence (NOUN)
Token 16: used (VERB)
Token 17: to (PART)
Token 18: display (VERB)
Token 19: all (DET)
Token 20: letters (NOUN)
```

```
print("Comparison of NLTK and spaCy POS Tags (first 20 tokens):")
print("-----------------------------------------------------")
print(f"{'Token':<15} {'NLTK Tag':<15} {'spaCy Tag':<15}")
print("-----------------------------------------------------")

# Ensure we don't go out of bounds for nltk_pos_tags
comparison_limit = min(len(spacy_doc), len(nltk_pos_tags), 20)

for i in range(comparison_limit):
    spacy_token = spacy_doc[i]
    nltk_word, nltk_tag = nltk_pos_tags[i]

    # NLTK words might include punctuation or be slightly different due to t
    # For direct comparison, we will use the word from nltk_pos_tags for con
    # but acknowledge spacy_token.text might be more accurate based on spacy
    print(f"{nltk_word:<15} {nltk_tag:<15} {spacy_token.pos_:<15}")
```

```
Comparison of NLTK and spaCy POS Tags (first 20 tokens):
--------------------------------------------------
Token           NLTK Tag        spaCy Tag
--------------------------------------------------
The             DT              DET
quick           JJ              ADJ
brown           NN              ADJ
fox             NN              NOUN
jumps           VBZ             VERB
over            IN              ADP
the             DT              DET
lazy            JJ              ADJ
dog             NN              NOUN
.               .               PUNCT
This            DT              PRON
is              VBZ             AUX
a               DT              DET
classic         JJ              ADJ
sentence        NN              NOUN
used            VBN             VERB
to              TO              PART
display         VB              VERB
all             DT              DET
letters         NNS             NOUN
```

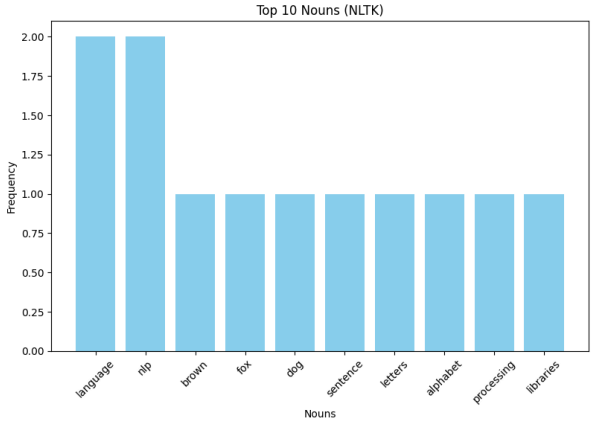
```
nltk_nouns = []
nltk_verbs = []

# Define NLTK POS tags for nouns and verbs
noun_tags = ['NN', 'NNS', 'NNP', 'NNPS']
verb_tags = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']

# Iterate through the NLTK POS tagged words and categorize them
for word, tag in nltk_pos_tags:
    if word.isalpha(): # Filter out punctuation and non-alphabetic tokens
        if tag in noun_tags:
            nltk_nouns.append(word.lower())
        elif tag in verb_tags:
            nltk_verbs.append(word.lower())

print("Extracted Nouns (first 10):", nltk_nouns[:10])
print("Total Nouns extracted:", len(nltk_nouns))
print("\nExtracted Verbs (first 10):", nltk_verbs[:10])
print("Total Verbs extracted:", len(nltk_verbs))
```

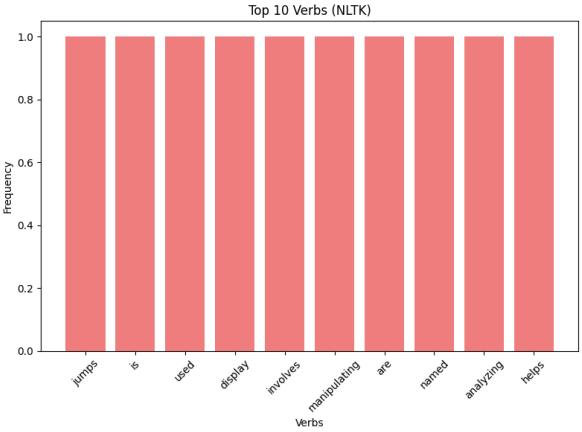
```
Extracted Nouns (first 10): ['brown', 'fox', 'dog', 'sentence', 'letters', 'a
Total Nouns extracted: 34

Extracted Verbs (first 10): ['jumps', 'is', 'used', 'display', 'involves', 'm
Total Verbs extracted: 17
```

```
from nltk.probability import FreqDist
import matplotlib.pyplot as plt

# Calculate frequency distribution for nouns
```

```
nltk_noun_freq = FreqDist(nltk_nouns)

# Calculate frequency distribution for verbs
nltk_verb_freq = FreqDist(nltk_verbs)

# Get the 10 most common nouns and verbs
top_10_nouns = nltk_noun_freq.most_common(10)
top_10_verbs = nltk_verb_freq.most_common(10)

print("\nTop 10 Nouns (NLTK):")
for word, freq in top_10_nouns:
    print(f"{word}: {freq}")

print("\nTop 10 Verbs (NLTK):")
for word, freq in top_10_verbs:
    print(f"{word}: {freq}")

# Prepare data for plotting
nouns_words = [word for word, freq in top_10_nouns]
nouns_counts = [freq for word, freq in top_10_nouns]

verbs_words = [word for word, freq in top_10_verbs]
verbs_counts = [freq for word, freq in top_10_verbs]

# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot top 10 nouns
axes[0].bar(nouns_words, nouns_counts, color='skyblue')
axes[0].set_xlabel('Nouns')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Top 10 Nouns (NLTK)')
axes[0].tick_params(axis='x', rotation=45)

# Plot top 10 verbs
axes[1].bar(verbs_words, verbs_counts, color='lightcoral')
axes[1].set_xlabel('Verbs')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Top 10 Verbs (NLTK)')
axes[1].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```

```
Top 10 Nouns (NLTK):
language: 2
nlp: 2
brown: 1
fox: 1
dog: 1
sentence: 1
letters: 1
alphabet: 1
processing: 1
libraries: 1

Top 10 Verbs (NLTK):
jumps: 1
is: 1
used: 1
display: 1
involves: 1
manipulating: 1
are: 1
named: 1
analyzing: 1
helps: 1
```






```
spacy_nouns = []
spacy_verbs = []

# Define spaCy POS tags for nouns and verbs
spacy_noun_tags = ['NOUN', 'PROPN']
spacy_verb_tags = ['VERB', 'AUX'] # AUX for auxiliary verbs like 'is', 'are' e

# Iterate through the spaCy processed document and categorize tokens
for token in spacy_doc:
    if token.text.isalpha(): # Filter out punctuation and non-alphabetic toker
        if token.pos_ in spacy_noun_tags:
            spacy_nouns.append(token.text.lower())
        elif token.pos_ in spacy_verb_tags:
            spacy_verbs.append(token.text.lower())

print("Extracted Nouns (first 10):", spacy_nouns[:10])
print("Total Nouns extracted:", len(spacy_nouns))
print("\nExtracted Verbs (first 10):", spacy_verbs[:10])
print("Total Verbs extracted:", len(spacy_verbs))
```

```
print("Total verbs extracted:", len(spacy_verbs))
```

```
Extracted Nouns (first 10): ['fox', 'dog', 'sentence', 'letters', 'alphabet',
Total Nouns extracted: 37

Extracted Verbs (first 10): ['jumps', 'is', 'used', 'display', 'involves', 'm
Total Verbs extracted: 18
```

```
from nltk.probability import FreqDist
import matplotlib.pyplot as plt

# Calculate frequency distribution for nouns
spacy_noun_freq = FreqDist(spacy_nouns)

# Calculate frequency distribution for verbs
spacy_verb_freq = FreqDist(spacy_verbs)

# Get the 10 most common nouns and verbs
top_10_spacy_nouns = spacy_noun_freq.most_common(10)
top_10_spacy_verbs = spacy_verb_freq.most_common(10)

print("\nTop 10 Nouns (spaCy):")
for word, freq in top_10_spacy_nouns:
    print(f"{word}: {freq}")

print("\nTop 10 Verbs (spaCy):")
for word, freq in top_10_spacy_verbs:
    print(f"{word}: {freq}")

# Prepare data for plotting
spacy_nouns_words = [word for word, freq in top_10_spacy_nouns]
spacy_nouns_counts = [freq for word, freq in top_10_spacy_nouns]

spacy_verbs_words = [word for word, freq in top_10_spacy_verbs]
spacy_verbs_counts = [freq for word, freq in top_10_spacy_verbs]

# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot top 10 nouns
axes[0].bar(spacy_nouns_words, spacy_nouns_counts, color='skyblue')
axes[0].set_xlabel('Nouns')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Top 10 Nouns (spaCy)')
axes[0].tick_params(axis='x', rotation=45)

# Plot top 10 verbs
axes[1].bar(spacy_verbs_words, spacy_verbs_counts, color='lightcoral')
axes[1].set_xlabel('Verbs')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Top 10 Verbs (spaCy)')
axes[1].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```

```
Top 10 Nouns (spaCy):
language: 2
nlp: 2
text: 2
fox: 1
dog: 1
sentence: 1
letters: 1
alphabet: 1
natural: 1
processing: 1

Top 10 Verbs (spaCy):
jumps: 1
is: 1
used: 1
display: 1
involves: 1
manipulating: 1
are: 1
named: 1
analyzing: 1
helps: 1
```