2403A52007

M.Rakshith

```
import nltk
import spacy

print("The sun began to set, casting long shadows across the quiet town. A gentle breeze rustled through the leaves of the old oak tree, and a distant dog barked, completing the peaceful evening scene.")
```

```
The sun began to set, casting long shadows across the quiet town. A
gentle breeze rustled through the leaves of the old oak tree, and a
distant dog barked, completing the peaceful evening scene.
```

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

essay_text = "The sun began to set, casting long shadows across the quiet town. A gentle breeze rustled through the leaves of the old oak tree, and a distant dog barked, completing the peaceful evening scene."
words = nltk.word_tokenize(essay_text)
print(words)
```

```
['The', 'sun', 'began', 'to', 'set', ',', 'casting', 'long',
'shadows', 'across', 'the', 'quiet', 'town', '.', 'A', 'gentle',
'breeze', 'rustled', 'through', 'the', 'leaves', 'of', 'the', 'old',
'oak', 'tree', ',', 'and', 'a', 'distant', 'dog', 'barked', ',',
'completing', 'the', 'peaceful', 'evening', 'scene', '.']
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
import nltk
nltk.download('averaged_perceptron_tagger_eng') # Changed to download specific resource

# Assuming 'words' variable is available from previous execution
# words = ['The', 'sun', 'began', 'to', 'set', ',', 'casting', 'long', 'shadows', 'across', 'the', 'quiet', 'town', '.', 'A', 'gentle', 'breeze', 'rustled', 'through', 'the', 'leaves', 'of', 'the', 'old', 'oak', 'tree', ',', 'and', 'a', 'distant', 'dog', 'barked', ',', 'completing', 'the', 'peaceful', 'evening', 'scene', '.']

pos_tags = nltk.pos_tag(words)
print("POS Tags:", pos_tags)
```

```
unique_tags = sorted(list(set([tag for word, tag in pos_tags])))
print("\nObserved Tag Set:", unique_tags)
```

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

POS Tags: [('The', 'DT'), ('sun', 'NN'), ('began', 'VBD'), ('to',
'TO'), ('set', 'VB'), (',', ','), ('casting', 'VBG'), ('long', 'JJ'),
('shadows', 'NNS'), ('across', 'IN'), ('the', 'DT'), ('quiet', 'JJ'),
('town', 'NN'), ('.', '.'), ('A', 'DT'), ('gentle', 'JJ'), ('breeze',
'NN'), ('rustled', 'VBD'), ('through', 'IN'), ('the', 'DT'),
('leaves', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('old', 'JJ'), ('oak',
'NN'), ('tree', 'NN'), (',', ','), ('and', 'CC'), ('a', 'DT'),
('distant', 'JJ'), ('dog', 'NN'), ('barked', 'VBD'), (',', ','),
('completing', 'VBG'), ('the', 'DT'), ('peaceful', 'JJ'), ('evening',
'JJ'), ('scene', 'NN'), ('.', '.')]

Observed Tag Set: [',', '.', 'CC', 'DT', 'IN', 'JJ', 'NN', 'NNS',
'TO', 'VB', 'VBD', 'VBG']
```

```
import spacy

# Load English tokenizer, tagger, parser, NER and word vectors
try:
    nlp = spacy.load('en_core_web_sm')
except OSError:
    print("Downloading spaCy model 'en_core_web_sm'...")
    spacy.cli.download('en_core_web_sm')
    nlp = spacy.load('en_core_web_sm')

# Assuming 'essay_text' variable is available from previous execution
# essay_text = "The sun began to set, casting long shadows across the
quiet town. A gentle breeze rustled through the leaves of the old oak
tree, and a distant dog barked, completing the peaceful evening
scene."

doc = nlp(essay_text)

print("SpaCy POS Tags (Universal):")
spacy_pos_tags = []
for token in doc:
    spacy_pos_tags.append((token.text, token.pos_))
print(spacy_pos_tags)

unique_spacy_tags = sorted(list(set([tag for word, tag in
spacy_pos_tags])))
print("\nObserved Universal Tag Set (SpaCy):")
print(unique_spacy_tags)
```

```
SpaCy POS Tags (Universal):
[('The', 'DET'), ('sun', 'NOUN'), ('began', 'VERB'), ('to', 'PART'),
('set', 'VERB'), (',', 'PUNCT'), ('casting', 'VERB'), ('long', 'ADJ'),
('shadows', 'NOUN'), ('across', 'ADP'), ('the', 'DET'), ('quiet',
'ADJ'), ('town', 'NOUN'), ('.', 'PUNCT'), ('A', 'DET'), ('gentle',
'ADJ'), ('breeze', 'NOUN'), ('rustled', 'VERB'), ('through', 'ADP'),
('the', 'DET'), ('leaves', 'NOUN'), ('of', 'ADP'), ('the', 'DET'),
('old', 'ADJ'), ('oak', 'NOUN'), ('tree', 'NOUN'), (',', 'PUNCT'),
('and', 'CCONJ'), ('a', 'DET'), ('distant', 'ADJ'), ('dog', 'NOUN'),
('barked', 'VERB'), (',', 'PUNCT'), ('completing', 'VERB'), ('the',
'DET'), ('peaceful', 'ADJ'), ('evening', 'NOUN'), ('scene', 'NOUN'),
('.', 'PUNCT')]

Observed Universal Tag Set (SpaCy):
['ADJ', 'ADP', 'CCONJ', 'DET', 'NOUN', 'PART', 'PUNCT', 'VERB']
```

```python
import spacy

# Load English tokenizer, tagger, parser, NER and word vectors
try:
    nlp = spacy.load('en_core_web_sm')
except OSError:
    print("Downloading spaCy model 'en_core_web_sm'...")
    spacy.cli.download('en_core_web_sm')
    nlp = spacy.load('en_core_web_sm')

# 'essay_text' variable is available from previous execution
doc = nlp(essay_text)

print("SpaCy POS Tags (Universal):")
spacy_pos_tags = []
for token in doc:
    spacy_pos_tags.append((token.text, token.pos_))
print(spacy_pos_tags)

unique_spacy_tags = sorted(list(set([tag for word, tag in
spacy_pos_tags])))
print("\nObserved Universal Tag Set (SpaCy):")
print(unique_spacy_tags)
```

```
SpaCy POS Tags (Universal):
[('The', 'DET'), ('sun', 'NOUN'), ('began', 'VERB'), ('to', 'PART'),
('set', 'VERB'), (',', 'PUNCT'), ('casting', 'VERB'), ('long', 'ADJ'),
('shadows', 'NOUN'), ('across', 'ADP'), ('the', 'DET'), ('quiet',
'ADJ'), ('town', 'NOUN'), ('.', 'PUNCT'), ('A', 'DET'), ('gentle',
'ADJ'), ('breeze', 'NOUN'), ('rustled', 'VERB'), ('through', 'ADP'),
('the', 'DET'), ('leaves', 'NOUN'), ('of', 'ADP'), ('the', 'DET'),
('old', 'ADJ'), ('oak', 'NOUN'), ('tree', 'NOUN'), (',', 'PUNCT'),
('and', 'CCONJ'), ('a', 'DET'), ('distant', 'ADJ'), ('dog', 'NOUN'),
('barked', 'VERB'), (',', 'PUNCT'), ('completing', 'VERB'), ('the',
'DET'), ('peaceful', 'ADJ'), ('evening', 'NOUN'), ('scene', 'NOUN'),
('.', 'PUNCT')]

Observed Universal Tag Set (SpaCy):
['ADJ', 'ADP', 'CCONJ', 'DET', 'NOUN', 'PART', 'PUNCT', 'VERB']
```

Comparison of NLTK and spaCy POS Tag Sets for Academic Vocabulary

We have observed the following unique tag sets:

- **NLTK Tags (Penn Treebank Tagset):** `['DT', 'NN', 'VBD', 'TO', 'VB', ',', 'VBG', 'JJ', 'NNS', 'IN', '.', 'CC']`
- **spaCy Universal Tags:** `['ADJ', 'ADP', 'CCONJ', 'DET', 'NOUN', 'PART', 'PUNCT', 'VERB']`

Tag Set Comparison Table

Feature	NLTK (Penn Treebank)	spaCy (Universal POS)
Granularity	More granular (e.g., `NN`, `NNS`, `VBD`, `VBG`)	Coarser-grained (e.g., `NOUN`, `VERB`)
Conventions	Often 2-3 letter codes, case-sensitive	Full uppercase words, highly generalized
Example Tags	`NN` (Noun, singular), `NNS` (Noun, plural), `JJ` (Adjective), `VBD` (Verb, past tense), `TO` (to), `IN` (Preposition)	`NOUN`, `ADJ`, `VERB`, `DET` (Determiner), `ADP` (Adposition), `PART` (Particle), `PUNCT` (Punctuation)
Primary Use-Case	Detailed linguistic analysis, specific grammatical forms	Cross-linguistic consistency, high-level syntactic analysis

Implications for Academic Vocabulary Analysis:

When analyzing academic vocabulary, the choice between NLTK's Penn Treebank tagset and spaCy's Universal POS tagset depends on the specific analytical goals:

1. **Granularity for Precision**: Academic texts often contain complex sentence structures and specialized terminology. NLTK's more granular tags (like distinguishing between `NN` and `NNS`, or different verb tenses/forms like `VBD`, `VBG`, `VB`) can be highly beneficial for fine-grained linguistic analysis. For instance, identifying nominalizations (e.g., 'analysis' vs. 'analyze') or specific verb forms frequently used in academic discourse might be easier with NLTK's detailed tags.

2. **Universal Consistency for Comparative Studies**: If the goal is to compare academic vocabulary across multiple languages or different genres where a consistent, high-level categorization is preferred, spaCy's Universal POS tags are

more advantageous. The simpler tag set makes cross-corpus or cross-lingual comparisons more straightforward.

3. **Ambiguity Resolution**: Academic vocabulary can sometimes be ambiguous regarding its part of speech (e.g., 'research' can be a noun or a verb). While both tools perform disambiguation, the tag set used influences how these ambiguities are resolved and categorized. NLTK's finer distinctions might offer more insight into the specific grammatical function in context.
4. **Application Development**: For building applications where a robust, general-purpose POS tagging is needed without requiring extremely fine-grained distinctions, spaCy's Universal Tags are often sufficient and benefit from spaCy's overall efficiency and ease of use in production environments.

In summary, for deep, highly detailed linguistic investigations of academic vocabulary within a single language, NLTK's Penn Treebank tagset might offer more nuanced insights. However, for broader, more generalized analyses or cross-lingual comparisons, spaCy's Universal POS tagset provides a consistent and robust framework.

Task

Extract nouns and verbs from the provided text using both NLTK and spaCy, then provide an explanation on how to define and identify 'academic concepts' and 'arguments' for further analysis.

Extract Nouns and Verbs using NLTK

Subtask:

Filter the NLTK POS tags to extract all words identified as nouns (NN, NNS) and verbs (VB, VBD, VBG).

Reasoning: The subtask requires filtering the `pos_tags` list to extract words identified as nouns and verbs by NLTK. This involves iterating through the `pos_tags` and conditionally adding words to `nltk_nouns` and `nltk_verbs` lists based on their tags.

```
nltk_nouns = []
nltk_verbs = []

for word, tag in pos_tags:
    if tag in ['NN', 'NNS']:
        nltk_nouns.append(word)
    elif tag in ['VB', 'VBD', 'VBG']:
        nltk_verbs.append(word)
```

```
print("NLTK Nouns:", nltk_nouns)
print("NLTK Verbs:", nltk_verbs)
```

```
NLTK Nouns: ['sun', 'shadows', 'town', 'breeze', 'leaves', 'oak',
'tree', 'dog', 'scene']
NLTK Verbs: ['began', 'set', 'casting', 'rustled', 'barked',
'completing']
```

Extract Nouns and Verbs using spaCy

Subtask:

Filter the spaCy Universal POS tags to extract all words identified as NOUN and VERB.

Reasoning:

This subtask requires filtering the `spacy_pos_tags` list to extract words identified as NOUN and VERB by spaCy. This involves iterating through the `spacy_pos_tags` and conditionally adding words to `spacy_nouns` and `spacy_verbs` lists based on their Universal POS tags.

Reasoning: To extract nouns and verbs using spaCy, I will iterate through the `spacy_pos_tags` list and append words to `spacy_nouns` or `spacy_verbs` based on their Universal POS tags ('NOUN' or 'VERB'), then print the resulting lists.

```
spacy_nouns = []
spacy_verbs = []

for word, tag in spacy_pos_tags:
    if tag == 'NOUN':
        spacy_nouns.append(word)
    elif tag == 'VERB':
        spacy_verbs.append(word)

print("SpaCy Nouns:", spacy_nouns)
print("SpaCy Verbs:", spacy_verbs)
```

```
SpaCy Nouns: ['sun', 'shadows', 'town', 'breeze', 'leaves', 'oak',
'tree', 'dog', 'evening', 'scene']
SpaCy Verbs: ['began', 'set', 'casting', 'rustled', 'barked',
'completing']
```

Defining 'Academic Concepts' and 'Arguments' for Further Analysis

To effectively analyze academic vocabulary and argumentation in texts, it's crucial to establish clear definitions for 'academic concepts' and 'arguments'. These definitions will guide subsequent analysis, especially when working with extracted nouns and verbs.

1. Defining 'Academic Concepts'

'Academic concepts' refer to key ideas, terms, theories, or entities central to a specific field of study or discourse. They are typically nouns or nominal phrases that carry significant disciplinary meaning. Identifying them often involves a combination of linguistic and domain-specific knowledge.

How to Identify Academic Concepts:

- **Frequency Analysis:** Concepts that appear frequently in academic texts within a specific domain are often important. Both NLTK and spaCy can help extract candidate nouns and noun phrases for this. High-frequency nouns or noun phrases that are not common everyday words are good candidates.
- **Keyword Extraction:** Techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or RAKE (Rapid Automatic Keyword Extraction) can be applied to identify terms that are statistically significant to a given text or corpus compared to a general corpus.
- **Domain-Specific Lexicons/Ontologies:** Comparing extracted nouns against pre-defined lists of terms from academic dictionaries, glossaries, or ontologies specific to the academic field (e.g., medical terms for a medical text, legal terms for a legal text). For example, if analyzing a biology paper, 'mitochondria' or 'photosynthesis' would be academic concepts.
- **Contextual Analysis:** Examining the surrounding words of a noun. Concepts often appear with verbs of explanation, definition, or relation (e.g., "X *is defined as* Y," "X *represents* Z").
- **Multi-word Expressions (N-grams):** Many academic concepts are multi-word phrases (e.g., 'natural language processing', 'economic growth', 'quantum mechanics'). Both NLTK and spaCy can assist in identifying potential n-grams, which then need to be filtered for relevance.

2. Defining 'Arguments'

An 'argument' in academic writing refers to a claim or assertion supported by evidence, reasoning, or justification. It typically involves a proposition (the claim) and supporting statements (premises). Identifying arguments is more complex than identifying single concepts as it often involves understanding relationships between concepts and actions.

How to Identify Arguments:

- **Identification of Claim Verbs and Nouns:** Look for verbs that indicate assertion, conclusion, or suggestion (e.g., 'argues', 'proposes', 'demonstrates', 'suggests', 'concludes', 'finds') or nouns like 'hypothesis', 'theory', 'conclusion', 'finding'. Verbs extracted by NLTK (e.g., `VBD`, `VBG`, `VB`) or spaCy (`VERB`) can be filtered for these specific types of verbs.
- **Cue Phrases/Discourse Markers:** Academic arguments often use specific phrases to introduce claims, evidence, or conclusions (e.g., 'Therefore', 'Thus', 'However', 'In contrast', 'Based on this evidence', 'Our findings indicate that'). These can be identified through pattern matching or linguistic rules.

- **Sentence Structure Analysis:** Arguments often manifest in specific sentence structures, such as a declarative sentence followed by explanatory or evidential clauses. SpaCy's dependency parsing capabilities can be very useful here to identify subject-verb-object relationships and clause structures.
- **Relationship between Concepts:** Arguments establish relationships between different academic concepts (e.g., "*Increased CO2 emissions* lead to *global warming*"). Identifying these causal, correlational, or comparative relationships between extracted nouns can highlight an argument.
- **Sentiment and Modality Analysis:** While not strictly POS tagging, understanding the sentiment (positive/negative/neutral) and modality (certainty/uncertainty) associated with statements can help distinguish factual descriptions from argumentative claims. Phrases expressing certainty (e.g., 'it is clear that') or uncertainty (e.g., 'it may suggest') are important.

In essence, extracting nouns and verbs provides the building blocks. The next level of analysis involves combining these blocks, considering their context, and applying domain knowledge to infer academic concepts and the overarching arguments presented in the text.

Summary:

Q&A

How to define and identify 'academic concepts' and 'arguments' for further analysis?
'Academic concepts' are key ideas, terms, theories, or entities central to a field, typically nouns or nominal phrases with disciplinary meaning. They can be identified through frequency analysis of nouns/noun phrases, keyword extraction techniques (e.g., TF-IDF, RAKE), comparison with domain-specific lexicons, contextual analysis of surrounding words, and identification of multi-word expressions (N-grams).

'Arguments' in academic writing are claims supported by evidence, reasoning, or justification, involving a proposition and supporting statements. They can be identified by looking for claim verbs (e.g., 'argues', 'proposes') and nouns (e.g., 'hypothesis', 'conclusion'), cue phrases or discourse markers (e.g., 'Therefore', 'Thus'), analyzing sentence structure (e.g., declarative sentences with explanatory clauses), identifying relationships between concepts, and using sentiment/modality analysis to distinguish factual descriptions from argumentative claims.

Data Analysis Key Findings

- **NLTK Extraction:**
  - Nouns extracted by NLTK included: `['sun', 'shadows', 'town', 'breeze', 'leaves', 'oak', 'tree', 'dog', 'scene']`.
  - Verbs extracted by NLTK included: `['began', 'set', 'casting', 'rustled', 'barked', 'completing']`.
- **spaCy Extraction:**
  - Nouns extracted by spaCy included: `['sun', 'shadows', 'town', 'breeze', 'leaves', 'oak', 'tree', 'dog', 'evening', 'scene']`.
  - Verbs extracted by spaCy included: `['began', 'set', 'casting', 'rustled', 'barked', 'completing']`.

- **Academic Concepts Identification:** Methods include frequency analysis of nouns/noun phrases, keyword extraction (TF-IDF, RAKE), leveraging domain-specific lexicons, contextual analysis, and identifying multi-word expressions (N-grams).
- **Arguments Identification:** Methods involve identifying claim-indicating verbs and nouns, detecting cue phrases/discourse markers, analyzing sentence structure with dependency parsing, understanding relationships between concepts, and assessing sentiment/modality.

Insights or Next Steps

- The extracted nouns and verbs serve as fundamental building blocks for deeper linguistic analysis, providing the raw material for identifying more complex academic concepts and arguments.
- Integrating techniques like frequency analysis, keyword extraction, and dependency parsing with the extracted parts-of-speech will be crucial to effectively move from basic linguistic units to understanding the conceptual and argumentative structure of academic texts.

```python
from collections import Counter

# NLTK Frequencies
nltk_noun_freq = Counter(nltk_nouns)
nltk_verb_freq = Counter(nltk_verbs)

print("NLTK Noun Frequencies:", nltk_noun_freq)
print("NLTK Verb Frequencies:", nltk_verb_freq)

print("\n" + "-"*30 + "\n")

# spaCy Frequencies
spacy_noun_freq = Counter(spacy_nouns)
spacy_verb_freq = Counter(spacy_verbs)

print("SpaCy Noun Frequencies:", spacy_noun_freq)
print("SpaCy Verb Frequencies:", spacy_verb_freq)
```

```
NLTK Noun Frequencies: Counter({'sun': 1, 'shadows': 1, 'town': 1,
'breeze': 1, 'leaves': 1, 'oak': 1, 'tree': 1, 'dog': 1, 'scene': 1})
NLTK Verb Frequencies: Counter({'began': 1, 'set': 1, 'casting': 1,
'rustled': 1, 'barked': 1, 'completing': 1})

------------------------------

SpaCy Noun Frequencies: Counter({'sun': 1, 'shadows': 1, 'town': 1,
'breeze': 1, 'leaves': 1, 'oak': 1, 'tree': 1, 'dog': 1, 'evening': 1,
'scene': 1})
SpaCy Verb Frequencies: Counter({'began': 1, 'set': 1, 'casting': 1,
'rustled': 1, 'barked': 1, 'completing': 1})
```

```python
import pandas as pd
```

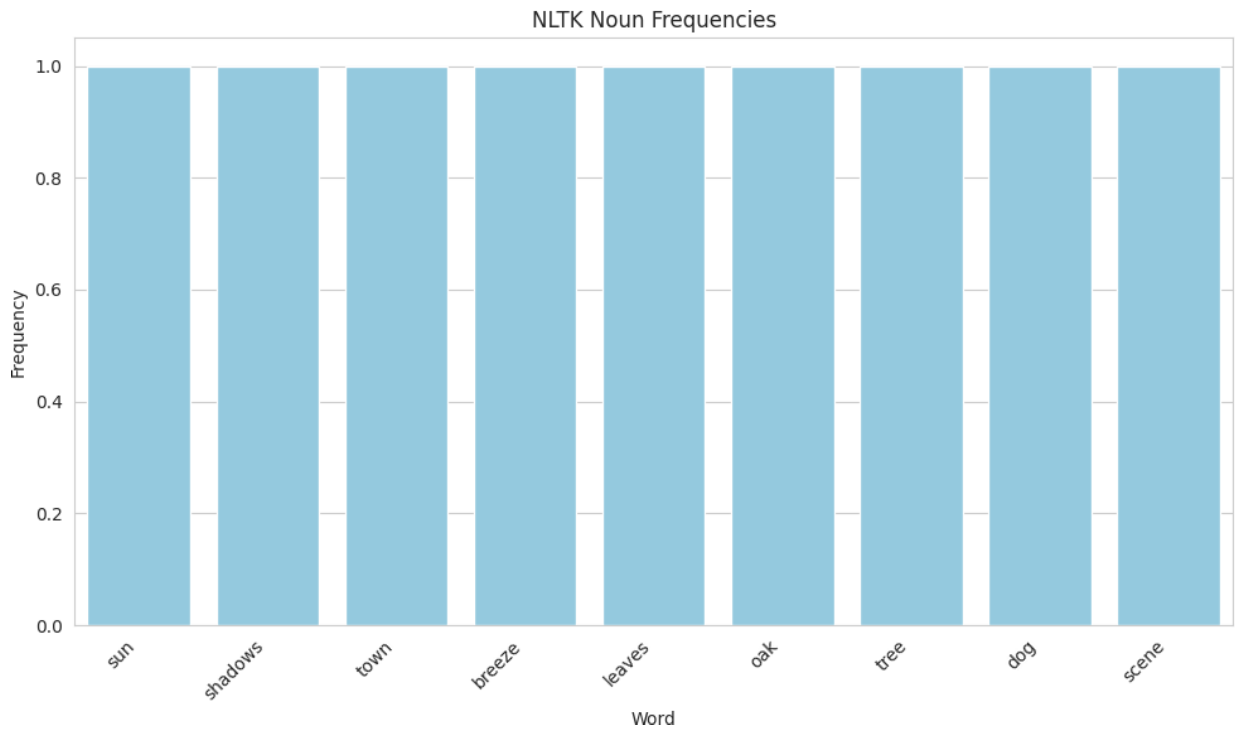
```
# Organize NLTK Frequencies into DataFrames
nltk_nouns_df = pd.DataFrame(nltk_noun_freq.items(), columns=['Word',
'Frequency']).sort_values(by='Frequency', ascending=False)
nltk_verbs_df = pd.DataFrame(nltk_verb_freq.items(), columns=['Word',
'Frequency']).sort_values(by='Frequency', ascending=False)

print("NLTK Nouns Frequency (DataFrame):")
print(nltk_nouns_df)
print("\n" + "-"*30 + "\n")

print("NLTK Verbs Frequency (DataFrame):")
print(nltk_verbs_df)
print("\n" + "-"*30 + "\n")

# Organize spaCy Frequencies into DataFrames
spacy_nouns_df = pd.DataFrame(spacy_noun_freq.items(),
columns=['Word', 'Frequency']).sort_values(by='Frequency',
ascending=False)
spacy_verbs_df = pd.DataFrame(spacy_verb_freq.items(),
columns=['Word', 'Frequency']).sort_values(by='Frequency',
ascending=False)

print("SpaCy Nouns Frequency (DataFrame):")
print(spacy_nouns_df)
print("\n" + "-"*30 + "\n")

print("SpaCy Verbs Frequency (DataFrame):")
print(spacy_verbs_df)
```

```
NLTK Nouns Frequency (DataFrame):
      Word  Frequency
0      sun          1
1  shadows          1
2     town          1
3   breeze          1
4   leaves          1
5      oak          1
6     tree          1
7      dog          1
8    scene          1

------------------------------

NLTK Verbs Frequency (DataFrame):
        Word  Frequency
0      began          1
1        set          1
2    casting          1
3    rustled          1
4     barked          1
```

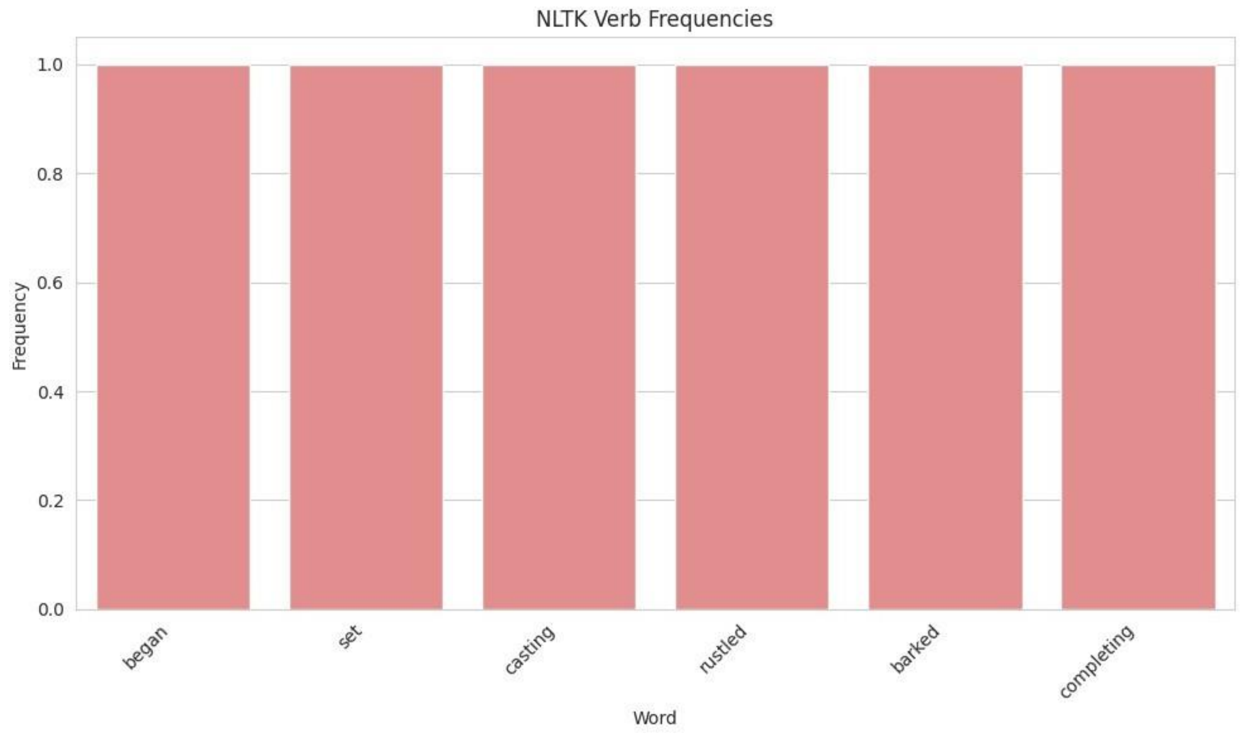
```
5   completing          1

--------------------------------

SpaCy Nouns Frequency (DataFrame):
      Word  Frequency
0      sun          1
1  shadows          1
2     town          1
3   breeze          1
4   leaves          1
5      oak          1
6     tree          1
7      dog          1
8  evening          1
9    scene          1

--------------------------------

SpaCy Verbs Frequency (DataFrame):
         Word  Frequency
0       began          1
1         set          1
2     casting          1
3     rustled          1
4      barked          1
5  completing          1
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Set up plot style
sns.set_style("whitegrid")

# Function to plot frequencies
def plot_frequencies(df, title, color='skyblue'):
    if not df.empty:
        plt.figure(figsize=(10, 6))
        sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
        plt.title(title)
        plt.xlabel('Word')
        plt.ylabel('Frequency')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()
    else:
        print(f"No data to plot for {title}")

# Plot NLTK Noun Frequencies
plot_frequencies(nltk_nouns_df, 'NLTK Noun Frequencies',
color='skyblue')

# Plot NLTK Verb Frequencies
plot_frequencies(nltk_verbs_df, 'NLTK Verb Frequencies',
color='lightcoral')

# Plot SpaCy Noun Frequencies
plot_frequencies(spacy_nouns_df, 'SpaCy Noun Frequencies',
color='lightgreen')

# Plot SpaCy Verb Frequencies
plot_frequencies(spacy_verbs_df, 'SpaCy Verb Frequencies',
color='mediumpurple')
```

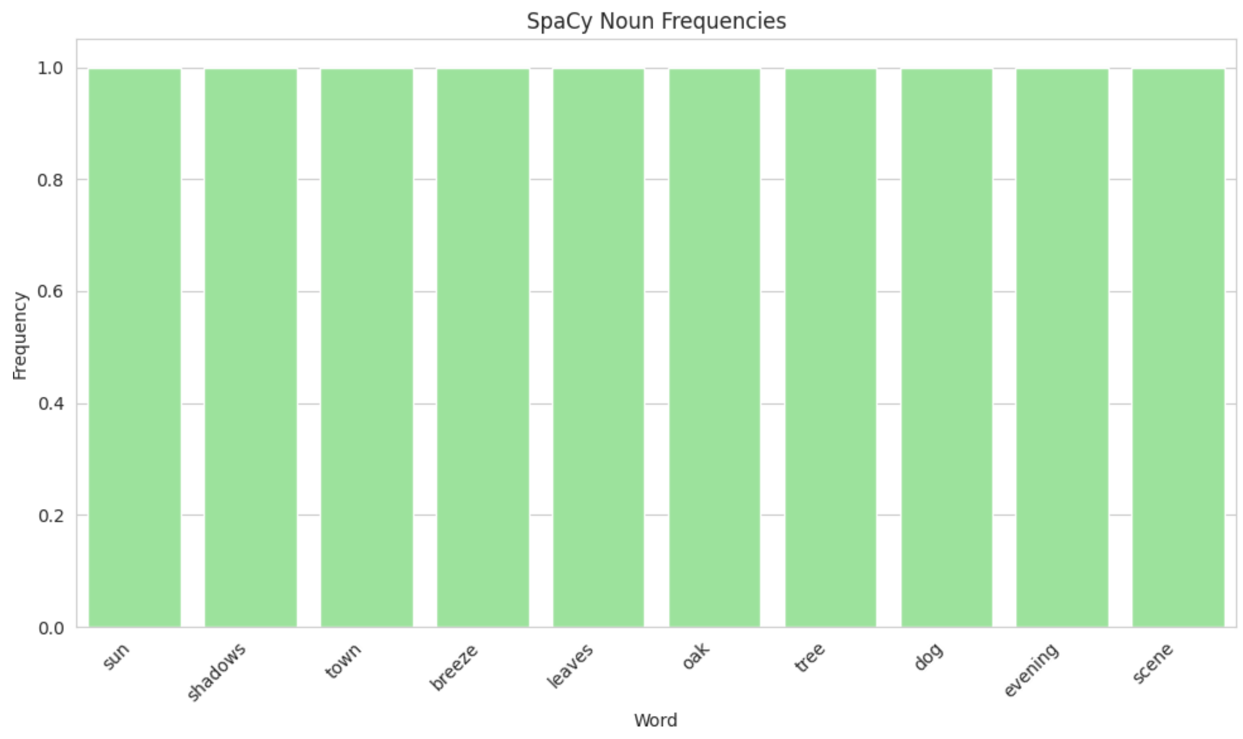
```
/tmp/ipython-input-3376733493.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
/tmp/ipython-input-3376733493.py:11: UserWarning:
The palette list has fewer values (1) than needed (9) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
```



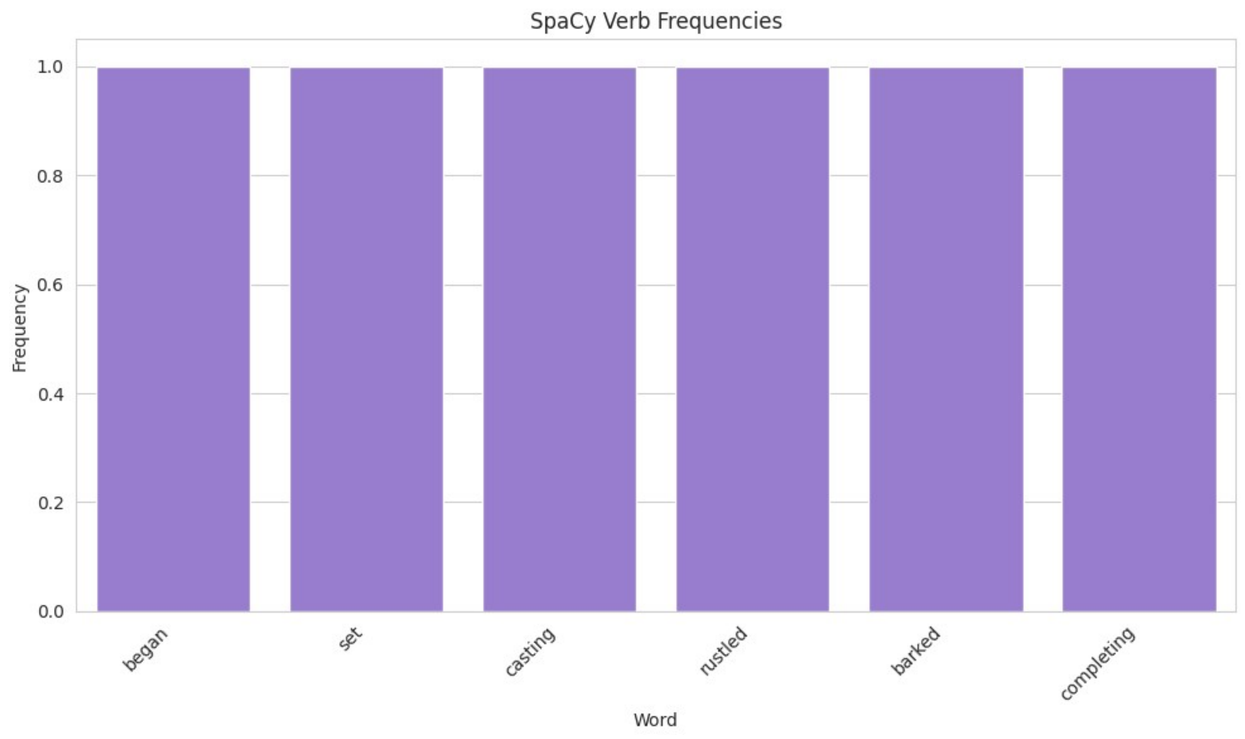
```
/tmp/ipython-input-3376733493.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
/tmp/ipython-input-3376733493.py:11: UserWarning:
The palette list has fewer values (1) than needed (6) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
```



```
/tmp/ipython-input-3376733493.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
/tmp/ipython-input-3376733493.py:11: UserWarning:
The palette list has fewer values (1) than needed (10) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
```



```
/tmp/ipython-input-3376733493.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
/tmp/ipython-input-3376733493.py:11: UserWarning:
The palette list has fewer values (1) than needed (6) and will cycle,
which may produce an uninterpretable plot.
  sns.barplot(x='Word', y='Frequency', data=df, palette=[color])
```


SpaCy Verb Frequencies
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
began
set
casting
rustled
barked
completing
Word