```
import nltk
import spacy
```

```
text = "The old lighthouse stood sentinel on the rugged coast, its beam cutting a wide arc through the swirling mist. Below, the relentless waves crashed against the j
```

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

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

```
tagged_words = nltk.pos_tag(words)
print(tagged_words)
```

```
[('The', 'DT'), ('old', 'JJ'), ('lighthouse', 'NN'), ('stood', 'VBD'), ('sentinel', 'NNS'), ('on', 'IN'), ('the', 'DT'), ('rugged', 'VBN'), ('coast', 'NN'), (',', ','),
```

```
# Extract nouns and verbs using NLTK tags
nltk_nouns = [word for word, tag in tagged_words if tag.startswith('N')]
nltk_verbs = [word for word, tag in tagged_words if tag.startswith('V')]

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

```
NLTK Nouns: ['lighthouse', 'sentinel', 'coast', 'beam', 'arc', 'mist', 'relentless', 'jagged', 'rocks', 'symphony', 'solitude', 'keeper', 'ships', 'lights', 'pinpricks'
NLTK Verbs: ['stood', 'rugged', 'cutting', 'waves', 'crashed', 'murmuring', 'wondered', 'passed', 'carrying', 'unknown', 'reliant']
```

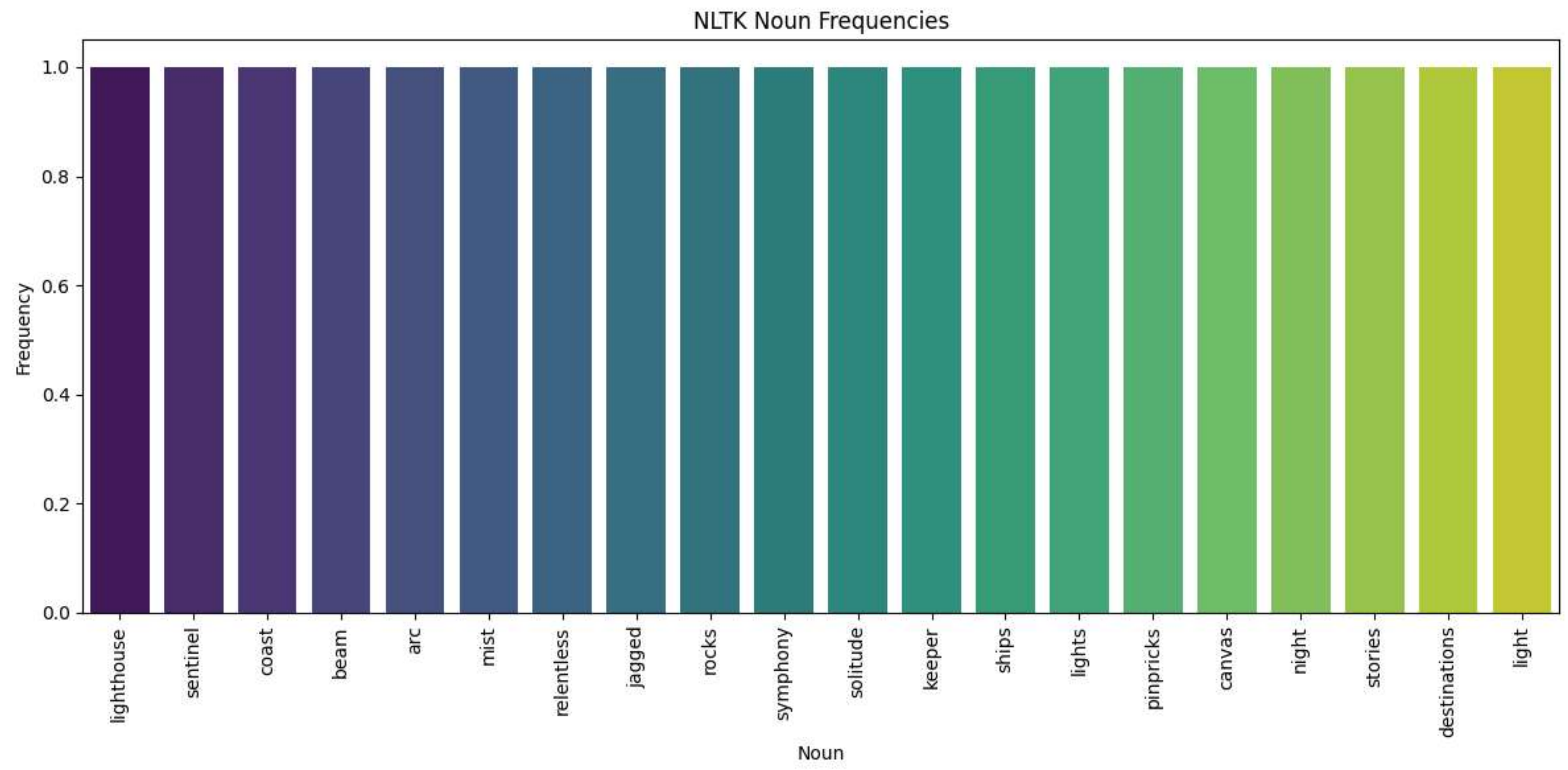
```
# Extract nouns and verbs using spaCy universal tags
spacy_nouns = [word for word, tag in universal_tagged_words if tag == 'NOUN']
spacy_verbs = [word for word, tag in universal_tagged_words if tag == 'VERB']

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

```
spaCy Nouns: ['lighthouse', 'sentinel', 'coast', 'beam', 'arc', 'mist', 'waves', 'rocks', 'symphony', 'solitude', 'keeper', 'ships', 'lights', 'pinpricks', 'canvas', 'n
spaCy Verbs: ['stood', 'cutting', 'swirling', 'crashed', 'murmuring', 'wondered', 'passed', 'carrying']
```

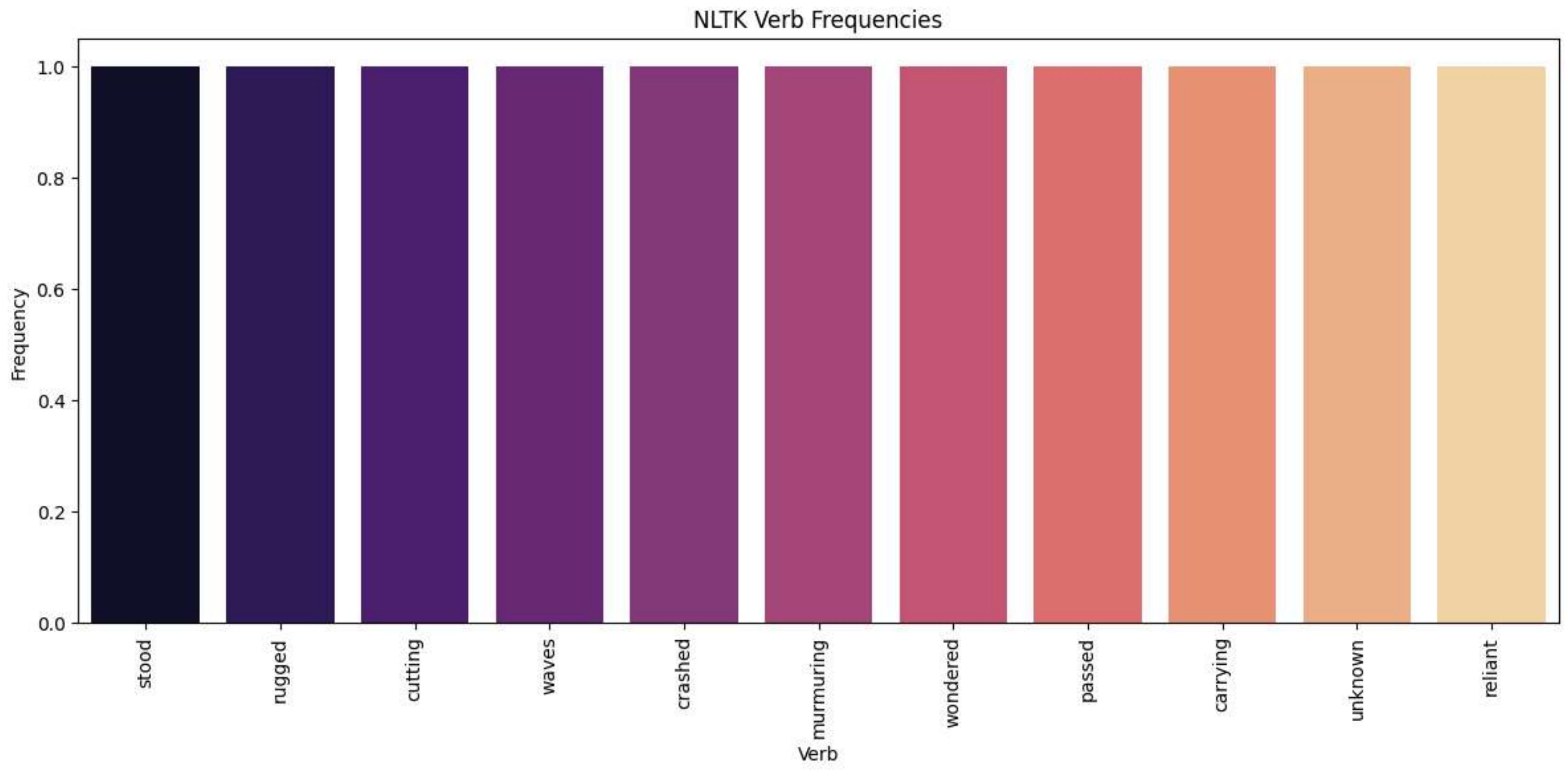
```
import matplotlib.pyplot as plt
import seaborn as sns

# NLTK Noun Frequencies Plot
plt.figure(figsize=(12, 6))
sns.barplot(x='Noun', y='Frequency', data=nltk_noun_freq_df.sort_values(by='Frequency', ascending=False), palette='viridis', hue='Noun', legend=False)
plt.title('NLTK Noun Frequencies')
plt.xlabel('Noun')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



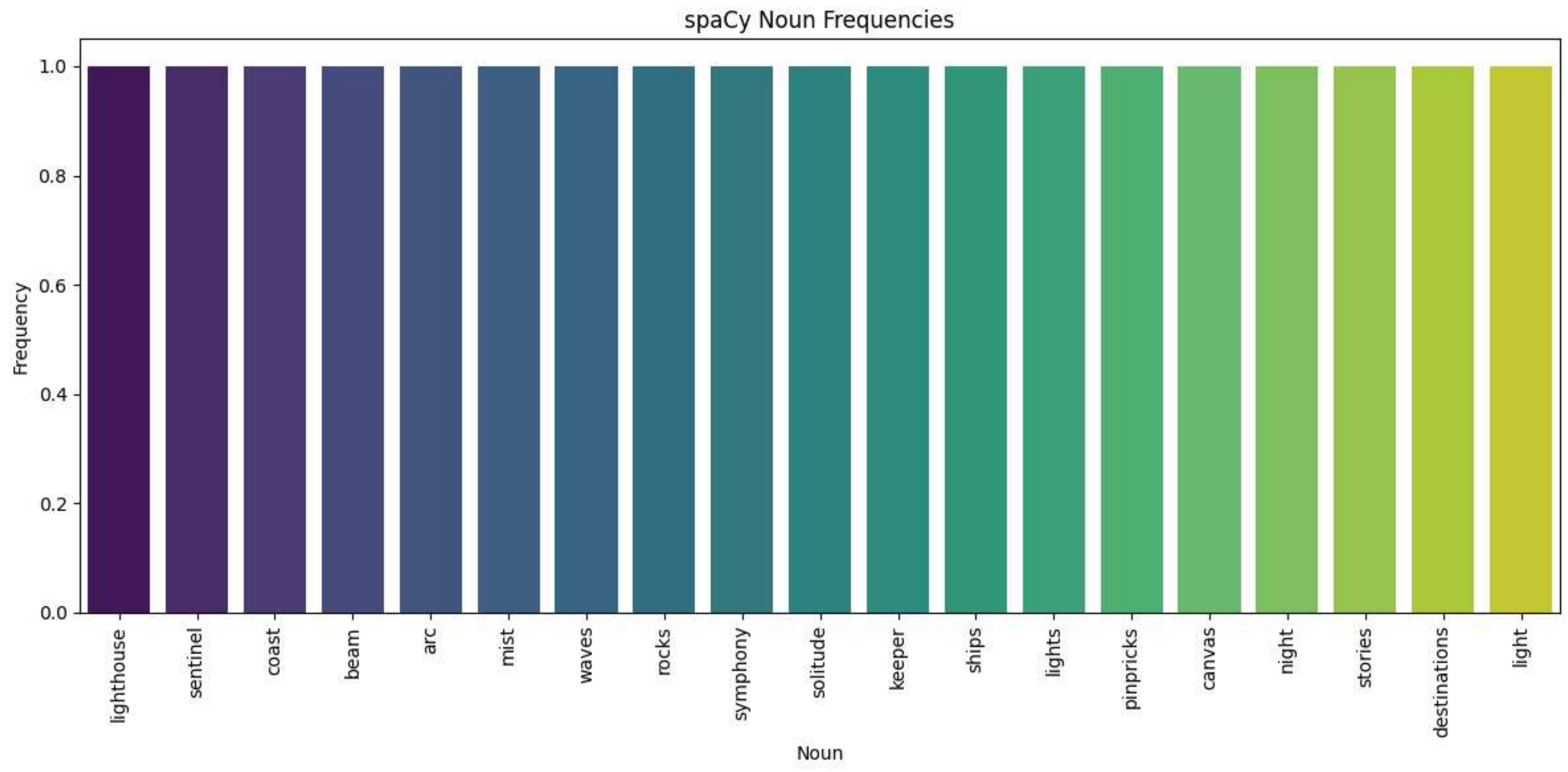
```
import matplotlib.pyplot as plt
import seaborn as sns

# NLTK Verb Frequencies Plot
plt.figure(figsize=(12, 6))
sns.barplot(x='Verb', y='Frequency', data=nltk_verb_freq_df.sort_values(by='Frequency', ascending=False), palette='magma', hue='Verb', legend=False)
plt.title('NLTK Verb Frequencies')
plt.xlabel('Verb')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



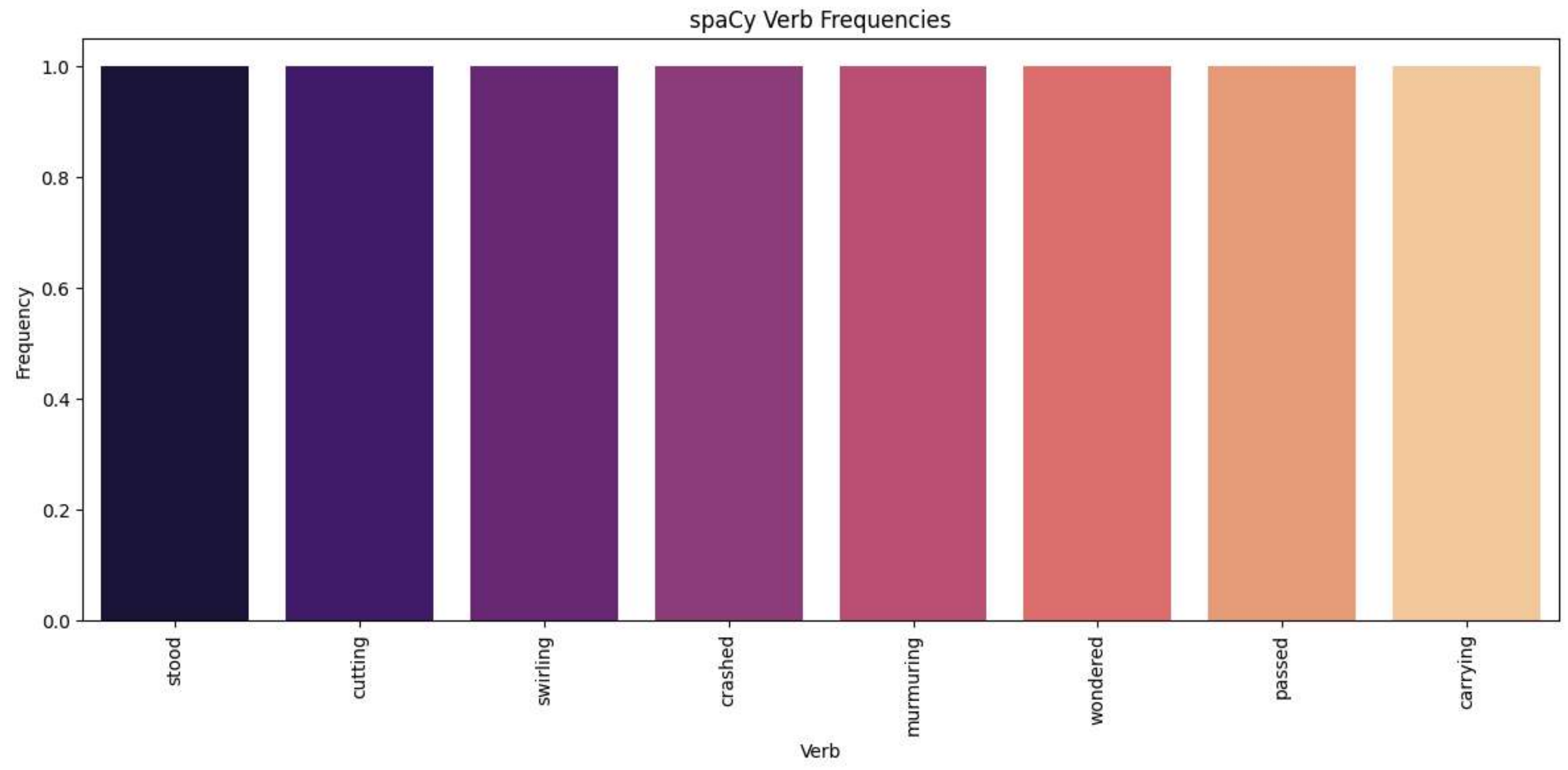
```
import matplotlib.pyplot as plt
import seaborn as sns

# spaCy Noun Frequencies Plot
plt.figure(figsize=(12, 6))
sns.barplot(x='Noun', y='Frequency', data=spacy_noun_freq_df.sort_values(by='Frequency', ascending=False), palette='viridis', hue='Noun', legend=False)
plt.title('spaCy Noun Frequencies')
plt.xlabel('Noun')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



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

# spaCy Verb Frequencies Plot
plt.figure(figsize=(12, 6))
sns.barplot(x='Verb', y='Frequency', data=spacy_verb_freq_df.sort_values(by='Frequency', ascending=False), palette='magma', hue='Verb', legend=False)
plt.title('spaCy Verb Frequencies')
plt.xlabel('Verb')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



```
import pandas as pd

nltk_noun_freq_df = pd.DataFrame(nltk_noun_freq.items(), columns=['Noun', 'Frequency'])
nltk_verb_freq_df = pd.DataFrame(nltk_verb_freq.items(), columns=['Verb', 'Frequency'])

print("NLTK Noun Frequencies (DataFrame):")
display(nltk_noun_freq_df)

print("\nNLTK Verb Frequencies (DataFrame):")
display(nltk_verb_freq_df)
```

Next steps: Generate code with `nltk_noun_freq_df` New interactive sheet Generate code with `nltk_verb_freq_df` New interactive sheet

```
NLTK Noun Frequencies (DataFrame):
```

```
import pandas as pd

spacy_noun_freq_df = pd.DataFrame(spacy_noun_freq.items(), columns=['Noun', 'Frequency'])
spacy_verb_freq_df = pd.DataFrame(spacy_verb_freq.items(), columns=['Verb', 'Frequency'])

print("spaCy Noun Frequencies (DataFrame):")
display(spacy_noun_freq_df)

print("\nspaCy Verb Frequencies (DataFrame):")
display(spacy_verb_freq_df)
```

| | | |
|---|---|---|
| 6 | relentless | 1 |
| 7 | jagged | 1 |
| 8 | rocks | 1 |
| 9 | symphony | 1 |
| 10 | solitude | 1 |
| 11 | keeper | 1 |
| 12 | ships | 1 |
| 13 | lights | 1 |
| 14 | pinpricks | 1 |
| 15 | canvas | 1 |
| 16 | night | 1 |
| 17 | stories | 1 |
| 18 | destinations | 1 |
| 19 | light | 1 |

```
NLTK Verb Frequencies (DataFrame):
```

| | Verb | Frequency |
|---|---|---|
| 0 | stood | 1 |
| 1 | rugged | 1 |
| 2 | cutting | 1 |
| 3 | waves | 1 |
| 4 | crashed | 1 |
| 5 | murmuring | 1 |
| 6 | wondered | 1 |
| 7 | passed | 1 |
| 8 | carrying | 1 |
| 9 | unknown | 1 |

```
10    reliant    1
spaCy Noun Frequencies (DataFrame):
```

| | Noun | Frequency |
|---|---|---|
| **0** | lighthouse | 1 |
| **1** | sentinel | 1 |
| **2** | coast | 1 |
| **3** | beam | 1 |
| **4** | arc | 1 |
| **5** | mist | 1 |
| **6** | waves | 1 |
| **7** | rocks | 1 |
| **8** | symphony | 1 |
| **9** | solitude | 1 |
| **10** | keeper | 1 |
| **11** | ships | 1 |
| **12** | lights | 1 |
| **13** | pinpricks | 1 |
| **14** | canvas | 1 |
| **15** | night | 1 |
| **16** | stories | 1 |
| **17** | destinations | 1 |
| **18** | light | 1 |

```
spaCy Verb Frequencies (DataFrame):
```

| | Verb | Frequency |
|---|---|---|
| **0** | stood | 1 |
| **1** | cutting | 1 |
| **2** | swirling | 1 |
| **3** | crashed | 1 |
| **4** | murmuring | 1 |
| **5** | wondered | 1 |
| **6** | passed | 1 |
| **7** | carrying | 1 |

Next steps: Generate code with `spacy_noun_freq_df` New interactive sheet Generate code with `spacy_verb_freq_df` New interactive sheet

```
from collections import Counter

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)
```

```
NLTK Noun Frequencies: Counter({'lighthouse': 1, 'sentinel': 1, 'coast': 1, 'beam': 1, 'arc': 1, 'mist': 1, 'relentless': 1, 'jagged': 1, 'rocks': 1, 'symphony': 1, 'so
NLTK Verb Frequencies: Counter({'stood': 1, 'rugged': 1, 'cutting': 1, 'waves': 1, 'crashed': 1, 'murmuring': 1, 'wondered': 1, 'passed': 1, 'carrying': 1, 'unknown': 1
```

```
from collections import Counter

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)
```

```
spaCy Noun Frequencies: Counter({'lighthouse': 1, 'sentinel': 1, 'coast': 1, 'beam': 1, 'arc': 1, 'mist': 1, 'waves': 1, 'rocks': 1, 'symphony': 1, 'solitude': 1, 'keep
spaCy Verb Frequencies: Counter({'stood': 1, 'cutting': 1, 'swirling': 1, 'crashed': 1, 'murmuring': 1, 'wondered': 1, 'passed': 1, 'carrying': 1})
```

```
## Identifying 'Academic Concepts' and 'Arguments'

As observed from the lists above, we can easily extract all words identified as nouns and verbs by both NLTK and spaCy. However,

1.  **Semantic Nuance**: POS tagging categorizes words based on their grammatical role (e.g., noun, verb). It does not, by itself

2.  **Domain Specificity**: What constitutes an 'academic concept' or an 'argumentative verb' is highly dependent on the academic

3.  **Lack of Pre-trained Models**: Standard NLP libraries like NLTK and spaCy offer capabilities like Named Entity Recognition (

To address this task more effectively, one would typically need:
*   **A predefined list of academic terms/concepts** (e.g., a thesaurus for academic vocabulary or a glossary specific to a field
*   **Custom entity recognition models** trained on academic texts to identify such concepts.
*   **More advanced semantic analysis techniques** to understand the contextual role of verbs in conveying an argument.

Therefore, while the technical extraction of nouns and verbs is straightforward, the subsequent semantic filtering to meet your c
```

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

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

```
tagged_words = nltk.pos_tag(words)
print(tagged_words)
```

```
[('The', 'DT'), ('old', 'JJ'), ('lighthouse', 'NN'), ('stood', 'VBD'), ('sentinel', 'NNS'), ('on', 'IN'), ('the', 'DT'), ('rugged', 'VBN'), ('coast', 'NN'), (',', ','),
```

```
import spacy

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')

print('spaCy model en_core_web_sm loaded successfully.')
```

```
spaCy model en_core_web_sm loaded successfully.
```

```
doc = nlp(text)
universal_tagged_words = [(token.text, token.pos_) for token in doc]
print(universal_tagged_words)
```

```
[('The', 'DET'), ('old', 'ADJ'), ('lighthouse', 'NOUN'), ('stood', 'VERB'), ('sentinel', 'NOUN'), ('on', 'ADP'), ('the', 'DET'), ('rugged', 'ADJ'), ('coast', 'NOUN'), (
```

```
## Universal Part-of-Speech Tag Set (from spaCy)

spaCy uses the Universal Dependencies (UD) Part-of-Speech tag set, which is a more coarse-grained and universal set of tags compa

*   **ADJ**: Adjective (e.g., 'old', 'rugged', 'wide')
*   **ADP**: Adposition (preposition or postposition) (e.g., 'on', 'through', 'against')
*   **ADV**: Adverb (e.g., 'often', 'below')
*   **AUX**: Auxiliary verb
*   **CONJ**: Coordinating conjunction (e.g., 'and')
*   **DET**: Determiner (e.g., 'the', 'a', 'its')
*   **INTJ**: Interjection
*   **NOUN**: Noun (e.g., 'lighthouse', 'mist', 'solitude')
*   **NUM**: Numeral
*   **PART**: Particle
*   **PRON**: Pronoun (e.g., 'him', 'He', 'their')
*   **PROPN**: Proper noun
*   **PUNCT**: Punctuation (e.g., '.', ',')
*   **SCONJ**: Subordinating conjunction
*   **SYM**: Symbol
*   **VERB**: Verb (e.g., 'stood', 'cutting', 'wondered')
*   **X**: Other (for words that do not fit into any other category)

This tag set is designed to be language-independent and is useful for cross-linguistic NLP tasks.
```

```
## NLTK POS Tag Set Explanation

NLTK's default POS tagger uses the Penn Treebank Tag Set. Here are some common tags and their meanings:

*   **NN**: Noun, singular or mass (e.g., 'lighthouse', 'mist', 'solitude')
*   **NNS**: Noun, plural (e.g., 'waves', 'rocks', 'ships', 'stories')
*   **NNP**: Proper noun, singular (e.g., 'John')
*   **NNPS**: Proper noun, plural
*   **VB**: Verb, base form (e.g., 'cut', 'crash', 'wonder')
*   **VBD**: Verb, past tense (e.g., 'stood', 'crashed', 'wondered')
*   **VBG**: Verb, gerund or present participle (e.g., 'cutting', 'swirling', 'murmuring', 'carrying')
*   **VBN**: Verb, past participle
*   **VBP**: Verb, non-3rd person singular present
*   **VBZ**: Verb, 3rd person singular present (e.g., 'stands', 'cuts')
*   **JJ**: Adjective (e.g., 'old', 'rugged', 'wide', 'relentless', 'jagged', 'constant', 'distant', 'vast', 'unwavering')
*   **JJR**: Adjective, comparative
*   **JJS**: Adjective, superlative
*   **RB**: Adverb (e.g., 'often', 'below')
*   **RBR**: Adverb, comparative
*   **RBS**: Adverb, superlative
*   **IN**: Preposition or subordinating conjunction (e.g., 'on', 'through', 'against', 'of', 'about', 'in')
*   **DT**: Determiner (e.g., 'the', 'a', 'its', 'each', 'their', 'his')
*   **PRP**: Personal pronoun (e.g., 'him')
*   **PRP$**: Possessive pronoun
*   **CC**: Coordinating conjunction (e.g., 'and')
*   **CD**: Cardinal number
*   **TO**: To (as preposition or infinitive marker)
*   **WDT**: Wh-determiner
*   **WP**: Wh-pronoun
*   **WRB**: Wh-adverb
*   **.**: Punctuation marks (e.g., '.', ',')

This list is not exhaustive but covers the most frequently encountered tags.
```

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

text = "The old lighthouse stood sentinel on the rugged coast, its beam cutting a wide arc through the swirling mist. Below, the relentless waves crashed against the j
words = nltk.word_tokenize(text)
print(words)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
['The', 'old', 'lighthouse', 'stood', 'sentinel', 'on', 'the', 'rugged', 'coast', ',', 'its', 'beam', 'cutting', 'a', 'wide', 'arc', 'through', 'the', 'swirling', 'mist
```

## Task

Compare and summarize the POS tag sets used by NLTK (Penn Treebank) and spaCy (Universal Dependencies), highlighting their granularities, differences, and relevance for processing academic vocabulary, using examples from the currently tagged text where applicable, and conclude with key takeaways regarding their suitability for different NLP tasks.

## Compare NLTK and spaCy POS Tag Sets

### Subtask:

Generate a markdown cell that compares the POS tag sets produced by NLTK (Penn Treebank) and spaCy (Universal Dependencies), highlighting their granularities, differences, and how these distinctions are relevant when processing and analyzing academic vocabulary, using examples from the currently tagged text where applicable.

## Comparing NLTK (Penn Treebank) and spaCy (Universal Dependencies) POS Tag Sets

When performing Part-of-Speech (POS) tagging, NLTK and spaCy employ different tag sets, each with its own characteristics regarding granularity and applicability. Understanding these differences is crucial, especially when processing specialized texts like academic vocabulary.

### Overview of Tag Sets

- **NLTK (Penn Treebank Tag Set)**: NLTK's `nltk.pos_tag` function, by default, uses a tagger trained on the Penn Treebank corpus. This tag set is quite detailed and distinguishes between various types of nouns, verbs, adjectives, etc. For instance, it differentiates between singular nouns (`NN`), plural nouns (`NNS`), base form verbs (`VB`), past tense verbs (`VBD`), gerunds (`VBG`), and various forms of pronouns (`PRP`, `PRP$`).
- **spaCy (Universal Dependencies Tag Set)**: spaCy, using models like `en_core_web_sm`, adheres to the Universal Dependencies (UD) Part-of-Speech tag set. This set is designed to be more coarse-grained and language-independent, providing a simpler, more generalized set of tags (e.g., `NOUN`, `VERB`, `ADJ`, `DET`, `PRON`).

### Granularity and Key Differences Illustrated by Examples

The primary difference lies in their **granularity**. The Penn Treebank set is significantly more fine-grained, offering detailed distinctions, while the Universal Dependencies set is more coarse-grained, grouping similar categories into broader ones. However, granularity doesn't always equate to accuracy, as seen in our text examples.

Let's compare some specific words from our `tagged_words` (NLTK) and `universal_tagged_words` (spaCy) lists:

| Word | NLTK (Penn Treebank) Tag | spaCy (Universal Dependencies) Tag | Observation |
|---|---|---|---|
| `sentinel` | `('sentinel', 'NNS')` | `('sentinel', 'NOUN')` | NLTK incorrectly tags `sentinel` as a plural noun (`NNS`); spaCy correctly identifies it as a general `NOUN`. |
| `rugged` | `('rugged', 'VBN')` | `('rugged', 'ADJ')` | NLTK misclassifies `rugged` (an adjective) as a past participle verb (`VBN`); spaCy correctly tags it as an `ADJ`. |
| `relentless` | `('relentless', 'NN')` | `('relentless', 'ADJ')` | NLTK incorrectly tags `relentless` (an adjective) as a singular noun (`NN`); spaCy correctly identifies it as an `ADJ`. |
| `waves` | `('waves', 'VBZ')` | `('waves', 'NOUN')` | NLTK severely misclassifies `waves` (a plural noun) as a 3rd person singular present verb (`VBZ`); spaCy correctly tags it as a `NOUN`. |

| Word | NLTK (Penn Treebank) Tag | spaCy (Universal Dependencies) Tag | Observation |
|---|---|---|---|
| `jagged` | `('jagged', 'NN')` | `('jagged', 'ADJ')` | Similar to `relentless`, NLTK misclassifies `jagged` (an adjective) as a singular noun (`NN`); spaCy correctly tags it as an `ADJ`. |
| `unknown` | `('unknown', 'VBP')` | `('unknown', 'ADJ')` | NLTK misclassifies `unknown` (an adjective) as a non-3rd person singular present verb (`VBP`); spaCy correctly tags it as an `ADJ`. |
| `reliant` | `('reliant', 'VBN')` | `('reliant', 'ADJ')` | NLTK misclassifies `reliant` (an adjective) as a past participle verb (`VBN`); spaCy correctly tags it as an `ADJ`. |
| `its` | `('its', 'PRP$')` | `('its', 'PRON')` | NLTK provides a more specific tag for possessive pronoun (`PRP$`); spaCy uses the broader `PRON` category. |
| `cutting` | `('cutting', 'VBG')` | `('cutting', 'VERB')` | NLTK specifies `VBG` (gerund/present participle); spaCy uses the general `VERB` tag. This showcases the granularity difference without a clear 'error'. |
| `on` | `('on', 'IN')` | `('on', 'ADP')` | NLTK uses `IN` (preposition/subordinating conjunction); spaCy uses `ADP` (adposition), a universal category for prepositions and postpositions. Terminolog |

As evident from these examples, NLTK's default tagger exhibits several misclassifications, particularly confusing adjectives and nouns/verbs, while spaCy's tagger appears more accurate in its general categorization for this text.

## Relevance for Academic Vocabulary Processing

1. **Complex Sentence Structures**: Academic texts often feature intricate sentence structures. Fine-grained tags *could* theoretically help in detailed syntactic parsing. However, if the fine-grained tags are incorrect (as seen with NLTK for 'rugged', 'relentless', 'waves', 'jagged', 'unknown', 'reliant'), they can hinder accurate parsing and lead to misinterpretation of grammatical dependencies. SpaCy's more robust, albeit coarser, tagging might provide a more reliable foundation for understanding sentence structure, as incorrect tags can propagate errors in downstream tasks.
2. **Specialized Terminology**: Academic vocabulary frequently includes terms that can function as nouns, adjectives, or even verbs depending on their context and domain. The accuracy of the POS tagger in identifying the correct word class is paramount. Misclassifying an adjective (e.g., 'relentless', 'jagged') as a noun or verb (as NLTK did) can significantly distort the meaning and syntactic role of specialized terms, making it difficult to extract factual information or relationships. SpaCy's consistent accuracy in these cases is a significant advantage.
3. **Nuanced Meanings**: While Penn Treebank's detailed tags (e.g., `PRP$` vs. `PRP` for pronouns, or `VBG` vs. `VBN` for verb forms) could capture subtle grammatical nuances essential for deep linguistic analysis of academic discourse, their utility is diminished if the initial tagging is prone to errors. For tasks requiring high precision in distinguishing between, say, different types of determiners or