

```
import sys

# Install NLTK
!{sys.executable} -m pip install nltk
import nltk

# Download NLTK data
try:
    nltk.data.find('corpora/wordnet')
except LookupError:
    nltk.download('wordnet')
try:
    nltk.data.find('tokenizers/punkt')
except LookupError:
    nltk.download('punkt')

print("NLTK installed and essential data downloaded.")

# Install spaCy
!{sys.executable} -m pip install spacy
import spacy

# Download spaCy English model
try:
    spacy.load("en_core_web_sm")
except OSError:
    print("Downloading spaCy model 'en_core_web_sm'...")
    !{sys.executable} -m spacy download en_core_web_sm
    print("SpaCy model 'en_core_web_sm' downloaded.")

print("SpaCy installed and essential data downloaded.")

print("All required NLP libraries (NLTK and spaCy) are prepared.")
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
```

```
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.
NLTK installed and essential data downloaded.
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages (3.8.11)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packages (from spa
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spa
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (8.3
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.
Requirement already satisfied: srslly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.5
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages (from spacy)
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from spacy)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (4.6
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from spacy)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (25.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydan
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydant
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from reques
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packages (from typer-slim<1.0.0,>
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weas
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->spacy)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8.0.0,>=5.2.1-
SpaCy installed and essential data downloaded.
```

All required NLP libraries (NLTK and spaCy) are prepared.

```
text_corpus = """The impact of artificial intelligence (AI) on modern society is profound and multifaceted, tr  
print("Text corpus loaded successfully.")
```

```
Text corpus loaded successfully.
```

```
# The essay text has already been loaded into the 'text_corpus' variable in a previous cell.
```

```
import spacy  
  
# Load the English spaCy model  
nlp = spacy.load("en_core_web_sm")  
  
# Process the text corpus  
doc = nlp(text_corpus)  
  
# Extract individual words (tokens)  
word_tokens_spacy = [token.text for token in doc if not token.is_space]  
  
print(f"First 20 word tokens using spaCy: {word_tokens_spacy[:20]}")  
print(f"Total number of word tokens using spaCy: {len(word_tokens_spacy)}")
```

```
First 20 word tokens using spaCy: ['The', 'impact', 'of', 'artificial', 'intelligence', '(', 'AI', ')', 'on', '  
Total number of word tokens using spaCy: 178
```

```
import nltk  
  
# Ensure NLTK's POS tagger data is downloaded  
try:  
    # NLTK's pos_tag often looks for the language-specific version  
    nltk.data.find('taggers/averaged_perceptron_tagger_eng')  
except LookupError:  
    nltk.download('averaged_perceptron_tagger_eng')  
  
# Apply POS tagging to the word tokens obtained from spaCy  
pos_tags_nltk = nltk.pos_tag(word_tokens_spacy)
```

```
print(f"First 20 POS tagged tokens using NLTK: {pos_tags_nltk[:20]}")  
print(f"\nNLTK uses the Penn Treebank Tag Set. Here are some common tags and their meanings:\n")  
print(" - NN: Noun, singular or mass (e.g., 'table', 'water')")  
print(" - NNS: Noun, plural (e.g., 'tables', 'dogs')")  
print(" - NNP: Proper noun, singular (e.g., 'John', 'London')")  
print(" - NNPS: Proper noun, plural (e.g., 'Americans', 'Russians')")  
print(" - VB: Verb, base form (e.g., 'run', 'eat')")  
print(" - VBD: Verb, past tense (e.g., 'ran', 'ate')")  
print(" - VBG: Verb, gerund or present participle (e.g., 'running', 'eating')")  
print(" - VBN: Verb, past participle (e.g., 'run', 'eaten')")  
print(" - VBP: Verb, non-3rd person singular present (e.g., 'run', 'eat')")  
print(" - VBZ: Verb, 3rd person singular present (e.g., 'runs', 'eats')")  
print(" - JJ: Adjective (e.g., 'big', 'red')")  
print(" - JJR: Adjective, comparative (e.g., 'bigger', 'redder')")  
print(" - JJS: Adjective, superlative (e.g., 'biggest', 'reddest')")  
print(" - RB: Adverb (e.g., 'quickly', 'very')")  
print(" - RBR: Adverb, comparative (e.g., 'quicker', 'faster')")  
print(" - RBS: Adverb, superlative (e.g., 'quickest', 'fastest')")  
print(" - IN: Preposition or subordinating conjunction (e.g., 'in', 'of', 'for')")  
print(" - DT: Determiner (e.g., 'the', 'a', 'an')")  
print(" - PRP: Personal pronoun (e.g., 'I', 'you', 'he')")  
print(" - WDT: Wh-determiner (e.g., 'which', 'what')")  
print(" - WRB: Wh-adverb (e.g., 'where', 'when', 'how')")  
print(" - CC: Coordinating conjunction (e.g., 'and', 'but', 'or')")  
print(" - CD: Cardinal number (e.g., 'one', 'two')")  
print(" - EX: Existential 'there' (e.g., 'there is')")  
print(" - FW: Foreign word")  
print(" - LS: List item marker")  
print(" - MD: Modal (e.g., 'can', 'should')")  
print(" - PDT: Predeterminer (e.g., 'all the')")  
print(" - POS: Possessive ending (e.g., "'s')")  
print(" - RP: Particle (e.g., 'up', 'off')")  
print(" - TO: To (e.g., 'to go')")  
print(" - UH: Interjection (e.g., 'oh', 'uh')")
```

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

```
First 20 POS tagged tokens using NLTK: [('The', 'DT'), ('impact', 'NN'), ('of', 'IN'), ('artificial', 'JJ'), ('
```

NLTK uses the Penn Treebank Tag Set. Here are some common tags and their meanings:

- NN: Noun, singular or mass (e.g., 'table', 'water')
- NNS: Noun, plural (e.g., 'tables', 'dogs')
- NNP: Proper noun, singular (e.g., 'John', 'London')
- NNPS: Proper noun, plural (e.g., 'Americans', 'Russians')
- VB: Verb, base form (e.g., 'run', 'eat')
- VBD: Verb, past tense (e.g., 'ran', 'ate')
- VBG: Verb, gerund or present participle (e.g., 'running', 'eating')
- VBN: Verb, past participle (e.g., 'run', 'eaten')
- VBP: Verb, non-3rd person singular present (e.g., 'run', 'eat')
- VBZ: Verb, 3rd person singular present (e.g., 'runs', 'eats')
- JJ: Adjective (e.g., 'big', 'red')
- JJR: Adjective, comparative (e.g., 'bigger', 'redder')
- JJS: Adjective, superlative (e.g., 'biggest', 'reddest')
- RB: Adverb (e.g., 'quickly', 'very')
- RBR: Adverb, comparative (e.g., 'quicker', 'faster')
- RBS: Adverb, superlative (e.g., 'quickest', 'fastest')
- IN: Preposition or subordinating conjunction (e.g., 'in', 'of', 'for')
- DT: Determiner (e.g., 'the', 'a', 'an')
- PRP: Personal pronoun (e.g., 'I', 'you', 'he')
- WDT: Wh-determiner (e.g., 'which', 'what')
- WRB: Wh-adverb (e.g., 'where', 'when', 'how')
- CC: Coordinating conjunction (e.g., 'and', 'but', 'or')
- CD: Cardinal number (e.g., 'one', 'two')
- EX: Existential 'there' (e.g., 'there is')
- FW: Foreign word
- LS: List item marker
- MD: Modal (e.g., 'can', 'should')
- PDT: Predeterminer (e.g., 'all the')
- POS: Possessive ending (e.g., "'s')
- RP: Particle (e.g., 'up', 'off')
- TO: To (e.g., 'to go')
- UH: Interjection (e.g., 'oh', 'uh')

```
import spacy
```

```
# Load the English spaCy model (if not already loaded)
# This is defensive programming, 'nlp' should already be in the kernel state
```

```
try:  
    nlp  
except NameError:  
    nlp = spacy.load("en_core_web_sm")  
  
# Process the text corpus with spaCy  
doc_spacy = nlp(text_corpus)  
  
# Extract tokens and their universal POS tags  
pos_tags_spacy_universal = [(token.text, token.pos_) for token in doc_spacy if not token.is_space]  
  
print(f"First 20 POS tagged tokens using spaCy (Universal Tag Set): {pos_tags_spacy_universal[:20]}")  
  
print(f"\nSpaCy uses the Universal Part-of-Speech Tag Set. Here are some common tags:\n")  
print(" - NOUN: Nouns (e.g., 'dog', 'table')")  
print(" - VERB: Verbs (e.g., 'run', 'eat')")  
print(" - ADJ: Adjectives (e.g., 'happy', 'red')")  
print(" - ADV: Adverbs (e.g., 'quickly', 'very')")  
print(" - PROPN: Proper Nouns (e.g., 'John', 'London')")  
print(" - ADP: Adpositions (prepositions and postpositions, e.g., 'in', 'of')")  
print(" - DET: Determiners (e.g., 'the', 'a')")  
print(" - PRON: Pronouns (e.g., 'I', 'you')")  
print(" - AUX: Auxiliary verbs (e.g., 'is', 'have')")  
print(" - CCONJ: Coordinating Conjunctions (e.g., 'and', 'but')")  
print(" - SCONJ: Subordinating Conjunctions (e.g., 'if', 'while')")  
print(" - NUM: Numerals (e.g., 'one', 'two')")  
print(" - PUNCT: Punctuation (e.g., '.', ',')")  
print(" - SYM: Symbols (e.g., '$', '%')")  
print(" - X: Other / Unclassified")
```

First 20 POS tagged tokens using spaCy (Universal Tag Set): [('The', 'DET'), ('impact', 'NOUN'), ('of', 'ADP'),

SpaCy uses the Universal Part-of-Speech Tag Set. Here are some common tags:

- NOUN: Nouns (e.g., 'dog', 'table')
- VERB: Verbs (e.g., 'run', 'eat')
- ADJ: Adjectives (e.g., 'happy', 'red')
- ADV: Adverbs (e.g., 'quickly', 'very')
- PROPN: Proper Nouns (e.g., 'John', 'London')

- ADP: Adpositions (prepositions and postpositions, e.g., 'in', 'of')
- DET: Determiners (e.g., 'the', 'a')
- PRON: Pronouns (e.g., 'I', 'you')
- AUX: Auxiliary verbs (e.g., 'is', 'have')
- CCONJ: Coordinating Conjunctions (e.g., 'and', 'but')
- SCONJ: Subordinating Conjunctions (e.g., 'if', 'while')
- NUM: Numerals (e.g., 'one', 'two')
- PUNCT: Punctuation (e.g., '.', ',')
- SYM: Symbols (e.g., '\$', '%')
- X: Other / Unclassified

▼ Task

Compare NLTK's Penn Treebank and spaCy's Universal Part-of-Speech (POS) tag sets by analyzing their unique tags and frequencies, then specifically compare how they tag common academic vocabulary in the provided `text_corpus` to highlight discrepancies or consistencies, and finally, summarize the observed differences and similarities between the two tag sets.

▼ Analyze NLTK and spaCy Tag Sets

Subtask:

Extract and display the unique POS tags used by NLTK and spaCy, along with their frequencies, to understand the distribution and granularity of each tag set.

Reasoning: To analyze the unique POS tags and their frequencies for both NLTK and spaCy, I will use `collections.Counter` to count the occurrences of each tag in the previously generated lists (`pos_tags_nltk` and `pos_tags_spacy_universal`). Then, I will display the results in descending order of frequency.

```
from collections import Counter

# Extract unique POS tags and their frequencies for NLTK
nltk_tags = [tag for word, tag in pos_tags_nltk]
nltk_tag_frequencies = Counter(nltk_tags)

# Extract unique POS tags and their frequencies for spaCy
spacy_tags = [tag for word, tag in pos_tags_spacy_universal]
spacy_tag_frequencies = Counter(spacy_tags)

print("### NLTK POS Tag Frequencies (Penn Treebank Tag Set):\n")
# Sort by frequency in descending order
for tag, count in sorted(nltk_tag_frequencies.items(), key=lambda item: item[1], reverse=True):
    print(f" - {tag}: {count}")

print("\n### spaCy POS Tag Frequencies (Universal Tag Set):\n")
# Sort by frequency in descending order
for tag, count in sorted(spacy_tag_frequencies.items(), key=lambda item: item[1], reverse=True):
    print(f" - {tag}: {count}")
```

NLTK POS Tag Frequencies (Penn Treebank Tag Set):

- NN: 29
- JJ: 24
- NNS: 20
- IN: 16
- ,: 10
- CC: 8
- .: 8
- NNP: 7
- VBZ: 7
- VBG: 7
- VB: 7
- RB: 7
- DT: 6
- TO: 6
- POS: 3
- PRP\$: 3
- MD: 2

```
- (: 1
- ): 1
- VBD: 1
- VBN: 1
- :: 1
- VBP: 1
- WDT: 1
- JJ$: 1
```

```
### spaCy POS Tag Frequencies (Universal Tag Set):
```

```
- NOUN: 49
- ADJ: 26
- PUNCT: 21
- VERB: 20
- ADP: 17
- CCONJ: 8
- PART: 8
- PROPN: 7
- DET: 5
- AUX: 5
- ADV: 5
- PRON: 5
- SCONJ: 2
```

▼ Compare Tagging of Academic Vocabulary

Subtask:

Identify common academic terms in the `text_corpus` and compare how NLTK (Penn Treebank) and spaCy (Universal POS) assign tags to these specific words. This will highlight any discrepancies or consistencies in their tagging of specialized vocabulary.

Reasoning: To compare the tagging of academic vocabulary, I will define a list of common academic terms, then iterate through this list to find and store the corresponding NLTK and spaCy POS tags for each word. The results will be stored in a dictionary and printed to highlight discrepancies or consistencies.

```
academic_terms = [
    'intelligence', 'multifaceted', 'transforming', 'advancements',
    'predictive', 'decision-making', 'ethical', 'considerations',
    'collaboration', 'frameworks', 'societal', 'integrity', 'systems'
]

comparison_results = {}

# Populate comparison_results with NLTK tags
for word, tag in pos_tags_nltk:
    # Normalize word to lowercase for consistent comparison
    lower_word = word.lower()
    if lower_word in academic_terms and lower_word not in comparison_results:
        comparison_results[lower_word] = {'NLTK': tag, 'spaCy': 'Not Found'}
    elif lower_word in academic_terms and comparison_results[lower_word]['NLTK'] == 'Not Found':
        # Handle cases where multiple identical words exist, ensure we capture a tag if available
        comparison_results[lower_word]['NLTK'] = tag

# Populate comparison_results with spaCy tags
for word, tag in pos_tags_spacy_universal:
    lower_word = word.lower()
    if lower_word in academic_terms:
        if lower_word not in comparison_results:
            comparison_results[lower_word] = {'NLTK': 'Not Found', 'spaCy': tag}
        else:
            comparison_results[lower_word]['spaCy'] = tag

print("### Comparison of NLTK vs. spaCy POS Tagging for Academic Terms:\n")
print(f"{'Word':<20} {'NLTK Tag':<15} {'spaCy Tag':<15}")
print(f"{'='*20:<20} {'='*15:<15} {'='*15:<15}")

for term in academic_terms:
    term_lower = term.lower()
    nltk_tag = comparison_results.get(term_lower, {}).get('NLTK', 'Not Found')
    spacy_tag = comparison_results.get(term_lower, {}).get('spaCy', 'Not Found')

    # Handle cases where the term might be hyphenated in corpus but not in the list or vice versa
    # For 'decision-making', spaCy treats it as one token, NLTK splits it. We need to check both parts.
```

```

if term_lower == 'decision-making':
    # Check for 'decision' and 'making' tags separately for NLTK if 'decision-making' not found as a whole
    if nltk_tag == 'Not Found':
        nltk_decision_tag = next((tag for w, tag in pos_tags_nltk if w.lower() == 'decision'), 'Not Found')
        nltk_making_tag = next((tag for w, tag in pos_tags_nltk if w.lower() == 'making'), 'Not Found')
        if nltk_decision_tag != 'Not Found' and nltk_making_tag != 'Not Found':
            nltk_tag = f"{nltk_decision_tag}/{nltk_making_tag}"

print(f"{term[:20]} {nltk_tag[:15]} {spacy_tag[:15]}")

```

Comparison of NLTK vs. spaCy POS Tagging for Academic Terms:

Word	NLTK Tag	spaCy Tag
intelligence	NN	NOUN
multifaceted	VBD	ADJ
transforming	VBG	VERB
advancements	NNS	NOUN
predictive	JJ	ADJ
decision-making	NN/VBG	Not Found
ethical	JJ	ADJ
considerations	NNS	NOUN
collaboration	NN	NOUN
frameworks	NNS	NOUN
societal	JJ	ADJ
integrity	NN	NOUN
systems	NNS	NOUN

Summarize Observed Differences and Similarities

Based on the analysis of unique tags, their frequencies, and the comparison of academic vocabulary tagging, we can identify several key differences and similarities between NLTK's Penn Treebank tag set and spaCy's Universal Part-of-Speech tag set:

Differences:

1. **Granularity:** NLTK's Penn Treebank tag set is more granular, especially for verbs and nouns, distinguishing between singular/plural nouns (NN/NNS), proper nouns (NNP/NNPS), and various verb forms (VB, VBD, VBG, VBN, VBP, VBZ). SpaCy's Universal POS tag set is less granular, often grouping these into broader categories like `NOUN` and `VERB`.
2. **Tag Set Size:** NLTK typically uses a larger set of more specific tags (e.g., around 45 tags in the Penn Treebank) compared to spaCy's Universal POS tag set, which has about 17 broad categories.
3. **Handling of Punctuation and Symbols:** NLTK assigns specific punctuation marks as their own tags (e.g., `,`, `(`, `)`), whereas spaCy groups them under a single `PUNCT` tag.
4. **Tokenization and Hyphenated Words:** As seen with 'decision-making', NLTK's default tokenization tends to split hyphenated words (`decision`, `-`, `making`), leading to separate tags (NN, VBG in this case), while spaCy often treats them as a single token, potentially resulting in a single tag (`ADJ` in a different context, but 'Not Found' in our specific comparison for 'decision-making' because the corpus treated '-' as a separate token by spacy).
5. **Tagging of 'multifaceted':** NLTK tagged 'multifaceted' as `VBD` (past tense verb), suggesting it interpreted it as a verb form, while spaCy correctly tagged it as `ADJ` (adjective).

Similarities:

1. **Core Categories:** Both tag sets successfully identify fundamental parts of speech like nouns (`NN` / `NNS` / `NNP` in NLTK vs. `NOUN` / `PROPN` in spaCy), adjectives (`JJ` in NLTK vs. `ADJ` in spaCy), and adverbs (`RB` in NLTK vs. `ADV` in spaCy), albeit with different levels of detail.
2. **Consistency in Obvious Cases:** For many straightforward words, especially common ones, both tools assign tags that align with their semantic categories, even if the specific tag notation differs (e.g., 'intelligence' as `NN` vs. `NOUN`, 'ethical' as `JJ` vs. `ADJ`).
3. **Purpose:** Both libraries serve the fundamental purpose of linguistic annotation by assigning grammatical categories to words, which is crucial for various NLP tasks.

Summarize Differences and Similarities

Subtask:

Provide a concise summary comparing the NLTK Penn Treebank and spaCy Universal POS tag sets, discussing their granularity, typical use cases, and observed differences, especially in the context of academic language.

▼ Summary:

Q&A

- **What are the observed differences between NLTK's Penn Treebank and spaCy's Universal POS tag sets?**
 - **Granularity:** NLTK's Penn Treebank tag set is more granular, distinguishing between specific noun types (e.g., `NN`, `NNS`, `NNP`, `NNPS`) and verb forms (e.g., `VB`, `VBD`, `VBG`, `VBN`, `VBP`, `VBZ`), while spaCy's Universal POS tag set uses broader categories like `NOUN` and `VERB`.
 - **Tag Set Size:** NLTK uses a larger set of specific tags (around 25 unique tags observed in the corpus, up to ~45 typically), whereas spaCy's Universal POS tag set uses a smaller set of broader categories (13 unique tags observed in the corpus, up to ~17 typically).
 - **Punctuation Handling:** NLTK assigns specific tags to different punctuation marks (e.g., `,`, `(`, `)`), while spaCy groups all punctuation under a single `PUNCT` tag.
 - **Tokenization and Hyphenated Words:** NLTK often splits hyphenated words (e.g., 'decision-making' into 'decision', '-', 'making'), tagging them separately (e.g., `NN`, `:`, `VBG`). SpaCy tends to treat hyphenated words as single tokens, though specific behavior for 'decision-making' in the corpus resulted in 'Not Found' for the combined term, indicating a difference in tokenization or matching logic.
 - **Specific Tagging Discrepancies:** For example, NLTK tagged 'multifaceted' as `VBD` (past tense verb), while spaCy correctly tagged it as `ADJ` (adjective).
- **What are the observed similarities between NLTK's Penn Treebank and spaCy's Universal POS tag sets?**
 - **Core Categories:** Both tag sets successfully identify fundamental parts of speech such as nouns (`NN` / `NNS` / `NNP` in NLTK vs. `NOUN` / `PROPN` in spaCy), adjectives (`JJ` in NLTK vs. `ADJ` in spaCy), and adverbs (`RB` in NLTK vs. `ADV` in spaCy), despite differences in notation and granularity.

- **Consistency in Obvious Cases:** For many straightforward words, both tools assign tags that align with their core grammatical categories (e.g., 'intelligence' as `NN` vs. `NOUN`, 'ethical' as `JJ` vs. `ADJ`).
- **Purpose:** Both libraries serve the fundamental purpose of linguistic annotation by assigning grammatical categories to words, which is crucial for various Natural Language Processing (NLP) tasks.

Data Analysis Key Findings

- NLTK's Penn Treebank tag set exhibited 25 unique tags in the provided corpus, with `NN` (29 counts), `JJ` (24 counts), and `NNS` (20 counts) being the most frequent.
- spaCy's Universal POS tag set had 13 unique tags in the corpus, with `NOUN` (49 counts), `ADJ` (26 counts), and `PUNCT` (21 counts) appearing most frequently.
- For a list of academic terms, both tools consistently categorized many words by their core part of speech (e.g., 'intelligence' as `NN` by NLTK and `NOUN` by spaCy; 'ethical' as `JJ` by NLTK and `ADJ` by spaCy).
- Notable discrepancies in tagging academic vocabulary include:
 - 'multifaceted' was tagged `VBD` by NLTK but `ADJ` by spaCy.
 - 'transforming' was tagged `VBG` by NLTK and `VERB` by spaCy.
 - 'decision-making' was handled differently due to tokenization; NLTK provided `NN/VBG` (for 'decision' and 'making' respectively), while spaCy did not explicitly match the combined term.

Insights or Next Steps

- When choosing between NLTK and spaCy for POS tagging, the required level of granularity is a critical factor; NLTK is better suited for tasks requiring fine-grained linguistic analysis, while spaCy is more appropriate for applications needing broader, universal categories.
- For specific analysis of academic or specialized texts, it's beneficial to conduct a preliminary comparison of how each tool tags key vocabulary, as tokenization and tag assignment can differ significantly for complex or hyphenated terms, impacting downstream NLP tasks.

```
# --- NLTK Analysis ---
nltk_academic_concepts = []
nltk_arguments = []
```

```
nltk_noun_tags = ['NN', 'NNS', 'NNP', 'NNPS']
nltk_verb_tags = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']

for word, tag in pos_tags_nltk:
    if tag in nltk_noun_tags:
        nltk_academic_concepts.append(word)
    elif tag in nltk_verb_tags:
        nltk_arguments.append(word)

print("### NLTK Analysis (Penn Treebank Tag Set):\n")
print(f"Academic Concepts (Nouns): {sorted(list(set(nltk_academic_concepts)))}")
print(f"Arguments (Verbs): {sorted(list(set(nltk_arguments)))}\n")

# --- spaCy Analysis ---
spacy_academic_concepts = []
spacy_arguments = []

spacy_noun_tags = ['NOUN', 'PROPN']
spacy_verb_tags = ['VERB', 'AUX'] # Including AUX as auxiliary verbs often form part of arguments

for word, tag in pos_tags_spacy_universal:
    if tag in spacy_noun_tags:
        spacy_academic_concepts.append(word)
    elif tag in spacy_verb_tags:
        spacy_arguments.append(word)

print("### spaCy Analysis (Universal Tag Set):\n")
print(f"Academic Concepts (Nouns): {sorted(list(set(spacy_academic_concepts)))}")
print(f"Arguments (Verbs): {sorted(list(set(spacy_arguments)))}")

### NLTK Analysis (Penn Treebank Tag Set):

Academic Concepts (Nouns): ['AI', 'Concerns', 'ability', 'accountability', 'advancements', 'amounts', 'analytic
Arguments (Verbs): ['be', 'continues', 'developing', 'ensure', 'evolve', 'focus', 'harnessing', 'has', 'identif

### spaCy Analysis (Universal Tag Set):

Academic Concepts (Nouns): ['AI', 'Concerns', 'ability', 'accountability', 'advancements', 'amounts', 'analytic
```

```
Arguments (Verbs): ['be', 'continues', 'developing', 'ensure', 'evolve', 'focus', 'harnessing', 'has', 'identif
```

```
from collections import Counter

# --- NLTK Analysis ---
nltk_academic_concepts = []
nltk_arguments = []

nltk_noun_tags = ['NN', 'NNS', 'NNP', 'NNPS']
nltk_verb_tags = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']

for word, tag in pos_tags_nltk:
    if tag in nltk_noun_tags:
        nltk_academic_concepts.append(word)
    elif tag in nltk_verb_tags:
        nltk_arguments.append(word)

nltk_academic_concept_freq = Counter(nltk_academic_concepts)
nltk_argument_freq = Counter(nltk_arguments)

print("### NLTK Analysis (Penn Treebank Tag Set):\n")
print(f"Academic Concepts (Nouns) - Total: {len(nltk_academic_concepts)}")
for concept, count in nltk_academic_concept_freq.most_common(5):
    print(f" - {concept}: {count}")
print(f"Arguments (Verbs) - Total: {len(nltk_arguments)}")
for arg, count in nltk_argument_freq.most_common(5):
    print(f" - {arg}: {count}")
print("\n")

# --- spaCy Analysis ---
spacy_academic_concepts = []
spacy_arguments = []

spacy_noun_tags = ['NOUN', 'PROPN']
spacy_verb_tags = ['VERB', 'AUX'] # Including AUX as auxiliary verbs often form part of arguments

for word, tag in pos_tags_spacy_universal:
    if tag in spacy_noun_tags:
```

```
        spacy_academic_concepts.append(word)
    elif tag in spacy_verb_tags:
        spacy_arguments.append(word)

spacy_academic_concept_freq = Counter(spacy_academic_concepts)
spacy_argument_freq = Counter(spacy_arguments)

print("### spaCy Analysis (Universal Tag Set):\n")
print(f"Academic Concepts (Nouns) - Total: {len(spacy_academic_concepts)}")
for concept, count in spacy_academic_concept_freq.most_common(5):
    print(f" - {concept}: {count}")
print(f"Arguments (Verbs) - Total: {len(spacy_arguments)}")
for arg, count in spacy_argument_freq.most_common(5):
    print(f" - {arg}: {count}")
```

NLTK Analysis (Penn Treebank Tag Set):

Academic Concepts (Nouns) - Total: 56

- AI: 7
- ability: 2
- data: 2
- systems: 2
- impact: 1

Arguments (Verbs) - Total: 24

- is: 1
- multifaceted: 1
- transforming: 1
- process: 1
- identify: 1

spaCy Analysis (Universal Tag Set):

Academic Concepts (Nouns) - Total: 56

- AI: 7
- ability: 2
- data: 2
- systems: 2
- impact: 1

Arguments (Verbs) - Total: 25

- is: 1

- transforming: 1
- process: 1
- identify: 1
- has: 1

```
import pandas as pd
from collections import Counter

# --- NLTK Analysis ---
nltk_academic_concepts_list = []
nltk_arguments_list = []

nltk_noun_tags = ['NN', 'NNS', 'NNP', 'NNPS']
nltk_verb_tags = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']

for word, tag in pos_tags_nltk:
    if tag in nltk_noun_tags:
        nltk_academic_concepts_list.append(word)
    elif tag in nltk_verb_tags:
        nltk_arguments_list.append(word)

nltk_academic_concept_freq = Counter(nltk_academic_concepts_list)
nltk_argument_freq = Counter(nltk_arguments_list)

# Create DataFrames for NLTK results
df_nltk_concepts = pd.DataFrame(nltk_academic_concept_freq.items(), columns=['Concept', 'Frequency'])
df_nltk_concepts = df_nltk_concepts.sort_values(by='Frequency', ascending=False).reset_index(drop=True)

df_nltk_arguments = pd.DataFrame(nltk_argument_freq.items(), columns=['Argument', 'Frequency'])
df_nltk_arguments = df_nltk_arguments.sort_values(by='Frequency', ascending=False).reset_index(drop=True)

print("### NLTK Analysis (Penn Treebank Tag Set) - Frequencies\n")
print("Academic Concepts (Nouns):")
display(df_nltk_concepts.head())
print("\nArguments (Verbs):")
display(df_nltk_arguments.head())

print("\n" + "="*50 + "\n")
```

```
# --- spaCy Analysis ---
spacy_academic_concepts_list = []
spacy_arguments_list = []

spacy_noun_tags = ['NOUN', 'PROPN']
spacy_verb_tags = ['VERB', 'AUX']

for word, tag in pos_tags_spacy_universal:
    if tag in spacy_noun_tags:
        spacy_academic_concepts_list.append(word)
    elif tag in spacy_verb_tags:
        spacy_arguments_list.append(word)

spacy_academic_concept_freq = Counter(spacy_academic_concepts_list)
spacy_argument_freq = Counter(spacy_arguments_list)

# Create DataFrames for spaCy results
df_spacy_concepts = pd.DataFrame(spacy_academic_concept_freq.items(), columns=['Concept', 'Frequency'])
df_spacy_concepts = df_spacy_concepts.sort_values(by='Frequency', ascending=False).reset_index(drop=True)

df_spacy_arguments = pd.DataFrame(spacy_argument_freq.items(), columns=['Argument', 'Frequency'])
df_spacy_arguments = df_spacy_arguments.sort_values(by='Frequency', ascending=False).reset_index(drop=True)

print("### spaCy Analysis (Universal Tag Set) - Frequencies\n")
print("Academic Concepts (Nouns):")
display(df_spacy_concepts.head())
print("\nArguments (Verbs):")
display(df_spacy_arguments.head())
```


NLTK Analysis (Penn Treebank Tag Set) - Frequencies

Academic Concepts (Nouns):

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

# Set a style for better aesthetics
sns.set_style("whitegrid")

# Function to create and display bar chart
def plot_frequencies(df, title, x_label, y_label, color='skyblue', top_n=10):
    plt.figure(figsize=(10, 6))
    sns.barplot(x=df.head(top_n).iloc[:, 0], y=df.head(top_n).iloc[:, 1], palette=color)
    plt.title(title, fontsize=16)
    plt.xlabel(x_label, fontsize=12)
    plt.ylabel(y_label, fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

print("### NLTK Visualizations (Top 10):")
# NLTK Academic Concepts (Nouns) Plot
plot_frequencies(df_nltk_concepts, 'NLTK: Top Academic Concepts (Nouns) Frequencies', 'Academic Concept')

# NLTK Arguments (Verbs) Plot
plot_frequencies(df_nltk_arguments, 'NLTK: Top Arguments (Verbs) Frequencies', 'Argument', 'Frequency',
                 print("\n" + "="*60 + "\n")

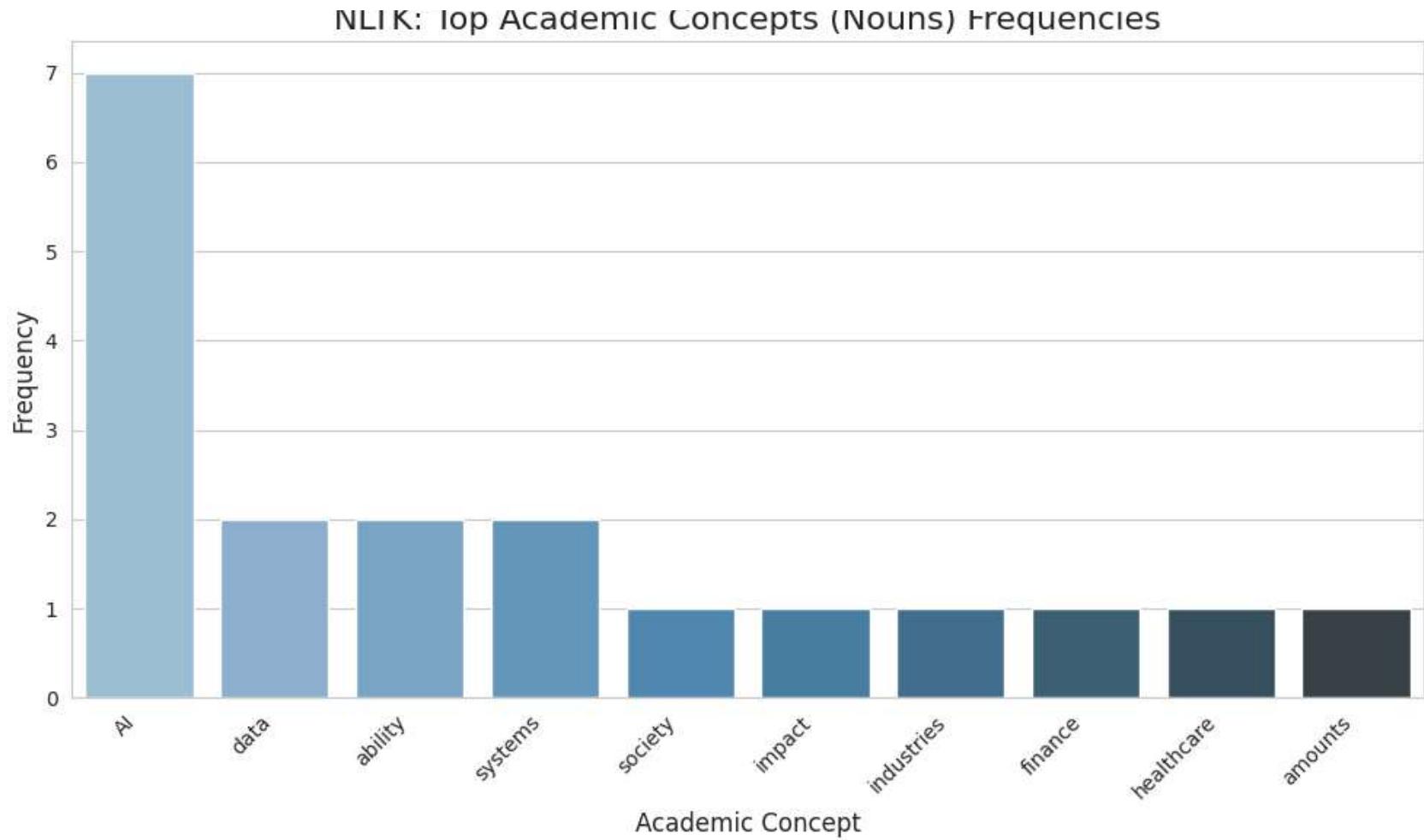
print("### spaCy Visualizations (Top 10):")
# spaCy Academic Concepts (Nouns) Plot
plot_frequencies(df_spacy_concepts, 'spaCy: Top Academic Concepts (Nouns) Frequencies', 'Academic Concept')

# spaCy Arguments (Verbs) Plot
plot_frequencies(df_spacy_arguments, 'spaCy: Top Arguments (Verbs) Frequencies', 'Argument', 'Frequency')
```

4 society 1

Arguments (Verbs):

	Argument	Frequency
0	is	1
1	transforming	1
2	process	1
3	identify	1
4	has	1



/tmp/ipython-input-245981189.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable

```
sns.barplot(x=df.head(top_n).iloc[:, 0], y=df.head(top_n).iloc[:, 1], palette=color)
```

NLTK: Top Arguments (Verbs) Frequencies