

Start coding or generate with AI.

▼ Task

Analyze an academic essay by performing Part-of-Speech (POS) tagging using both NLTK and spaCy, comparing their results, and identifying the most frequent nouns and verbs for insights, followed by visualization of these frequencies and a comprehensive discussion of the findings.

▼ Install and Download Resources

Subtask:

Install the `nltk` and `spaCy` libraries, and download necessary NLTK data (like 'punkt' and 'averaged_perceptron_tagger') and a spaCy language model (e.g., 'en_core_web_sm').

Reasoning: Install the required libraries `nltk` and `spaCy` using pip, then download the specified NLTK data and spaCy language model.

```
import sys

# Install nltk and spacy
!{sys.executable} -m pip install nltk spacy

import nltk
import spacy

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

# Download spaCy language model
!{sys.executable} -m spacy download en_core_web_sm

print("NLTK and spaCy libraries installed and resources downloaded successfully.")
```

```
..12/dist-packages (3.9.1)
3.12/dist-packages (3.8.11)
3.12/dist-packages (from nltk) (8.3.1)
in3.12/dist-packages (from nltk) (1.5.3)
lib/python3.12/dist-packages (from nltk) (2025.11.3)
..12/dist-packages (from nltk) (4.67.1)
 /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
 /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
/usr/local/lib/python3.12/dist-packages (from spacy) (1.0.15)
cal/lib/python3.12/dist-packages (from spacy) (2.0.13)
local/lib/python3.12/dist-packages (from spacy) (3.0.12)
cal/lib/python3.12/dist-packages (from spacy) (8.3.10)
ocal/lib/python3.12/dist-packages (from spacy) (1.1.3)
cal/lib/python3.12/dist-packages (from spacy) (2.5.2)
r/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
ocal/lib/python3.12/dist-packages (from spacy) (0.4.3)
sr/local/lib/python3.12/dist-packages (from spacy) (0.21.1)
b/python3.12/dist-packages (from spacy) (2.0.2)
```

```
r/local/lib/python3.12/dist-packages (from spacy) (2.32.4)
=1.7.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.12.3)
in3.12/dist-packages (from spacy) (3.1.6)
ython3.12/dist-packages (from spacy) (75.2.0)
lib/python3.12/dist-packages (from spacy) (25.0)
/local/lib/python3.12/dist-packages (from pydantic!=1.8,!>1.8.1,<3.0.0,>=1.7.4->spacy) (0.7.0)
local/lib/python3.12/dist-packages (from pydantic!=1.8,!>1.8.1,<3.0.0,>=1.7.4->spacy) (2.41.4)
/usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!>1.8.1,<3.0.0,>=1.7.4->spacy) (4.15.0)
sr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!>1.8.1,<3.0.0,>=1.7.4->spacy) (0.4.2)
sr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4.4)
/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.11)
al/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.5.0)
al/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2026.1.4)
al/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.3.3)
sr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (0.1.5)
/usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (0.23.0)
sr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (7.5.0)
lib/python3.12/dist-packages (from jinja2>spacy) (3.0.3)
3.12/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.4.2->spacy) (2.0.1)
```

p.

[ses/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-any.whl](#) (12.8 MB)
1.2 MB/s eta 0:00:00

```
start Python in
by selecting the
successfully.
```

Reasoning: The previous step successfully installed the necessary libraries and downloaded the required data. The next logical step, according to the overall task of analyzing an academic essay, is to define the academic essay text that will be used for analysis. This step prepares the input data for subsequent POS tagging.

```
essay_text = """The rapid advancements in artificial intelligence (AI) have ignited a profound de
print("Academic essay content loaded successfully.")

Academic essay content loaded successfully.
```

Reasoning: With the essay text loaded, the next step is to perform Part-of-Speech (POS) tagging using NLTK, which involves tokenizing the text into words and then applying the NLTK POS tagger to identify the grammatical category of each word.

```
from nltk.tokenize import word_tokenize
import nltk

# Ensure all necessary NLTK data is downloaded for tokenization and tagging
nltk.download('punkt') # For robustness, as it might be needed for word_tokenize
nltk.download('averaged_perceptron_tagger') # Needed for nltk.pos_tag
nltk.download('punkt_tab') # Needed for word_tokenize, as seen in previous error

# Tokenize the essay text using NLTK
...
```

```

nltk_tokens = word_tokenize(essay_text)

# Perform POS tagging using NLTK
nltk_pos_tags = nltk.pos_tag(nltk_tokens)

print("NLTK POS tagging completed successfully. Displaying first 10 tags:")
print(nltk_pos_tags[:10])

```

NLTK POS tagging completed successfully. Displaying first 10 tags:
[('The', 'DT'), ('rapid', 'JJ'), ('advancements', 'NNS'), ('in', 'IN'), ('artificial', 'JJ'), ('in', 'IN')]
[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] Package punkt_tab is already up-to-date!

Reasoning: The previous code failed because the 'punkt_tab' resource required by NLTK's `word_tokenize` function was not downloaded. I will add `nltk.download('punkt_tab')` to ensure this resource is available before tokenization.

```

from nltk.tokenize import word_tokenize
import nltk

# Ensure all necessary NLTK data is downloaded for tokenization and tagging
nltk.download('punkt') # For word_tokenize, for robustness
nltk.download('averaged_perceptron_tagger') # For pos_tag, for robustness
nltk.download('punkt_tab') # Needed for word_tokenize, as seen in previous error
nltk.download('averaged_perceptron_tagger_eng') # Explicitly download as suggested by error

# Tokenize the essay text using NLTK
ltk_tokens = word_tokenize(essay_text)

# Perform POS tagging using NLTK
ltk_pos_tags = nltk.pos_tag(ltk_tokens)

print("NLTK POS tagging completed successfully. Displaying first 10 tags:")
print(ltk_pos_tags[:10])

```

NLTK POS tagging completed successfully. Displaying first 10 tags:
[('The', 'DT'), ('rapid', 'JJ'), ('advancements', 'NNS'), ('in', 'IN'), ('artificial', 'JJ'), ('in', 'IN')]
[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] Package punkt_tab is already up-to-date!
[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!

Reasoning: The previous code failed because the `nltk.pos_tag` function could not find the 'averaged_perceptron_tagger_eng' resource, even though 'averaged_perceptron_tagger' was supposedly

downloaded earlier. This step will ensure both 'punkt_tab' (from the previous error) and 'averaged_perceptron_tagger_eng' are explicitly downloaded within this cell, along with 'punkt' and 'averaged_perceptron_tagger' for robustness, before performing NLTK POS tagging.

```
from nltk.tokenize import word_tokenize
import nltk

# Ensure all necessary NLTK data is downloaded for tokenization and tagging
nltk.download('punkt') # Already downloaded, but for robustness
nltk.download('averaged_perceptron_tagger') # Already downloaded, but for robustness
nltk.download('punkt_tab') # Needed for word_tokenize
nltk.download('averaged_perceptron_tagger_eng') # Explicitly download as suggested by error

# Tokenize the essay text using NLTK
ltk_tokens = word_tokenize(essay_text)

# Perform POS tagging using NLTK
ltk_pos_tags = nltk.pos_tag(ltk_tokens)

print("NLTK POS tagging completed successfully. Displaying first 10 tags:")
print(ltk_pos_tags[:10])

NLTK POS tagging completed successfully. Displaying first 10 tags:
[('The', 'DT'), ('rapid', 'JJ'), ('advancements', 'NNS'), ('in', 'IN'), ('artificial', 'JJ'), ('in', 'IN'), ('the', 'DT'), ('most', 'JJ'), ('common', 'NN'), ('of', 'IN'), ('these', 'PRP$'), ('are', 'VBP'), ('the', 'DT'), ('so-called', 'PRP$'), ('natural', 'JJ'), ('language', 'NN'), ('models', 'NNS'), ('that', 'PRP$'), ('have', 'VBD'), ('been', 'VBN'), ('trained', 'VBN'), ('on', 'IN'), ('large', 'JJ'), ('corpora', 'NNS'), ('and', 'CC'), ('then', 'PRP$'), ('they', 'PRP$'), ('are', 'VBD'), ('used', 'VBN'), ('to', 'TO'), ('map', 'VBZ'), ('out', 'VBN'), ('the', 'DT'), ('meaning', 'NN'), ('of', 'IN'), ('new', 'JJ'), ('words', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('context', 'NN'), ('in', 'IN'), ('which', 'PRP$'), ('they', 'PRP$'), ('have', 'VBD'), ('been', 'VBN'), ('seen', 'VBN'), ('before', 'RB'), ('.', 'PUNKT')]
```

Reasoning: The previous code resulted in a `NameError` because `nltk_pos_tags` was used in the `print` statement, but the POS tags were assigned to `ltk_pos_tags`. Additionally, the `nltk.pos_tag` function was incorrectly using `nltk_tokens` instead of `ltk_tokens`. The fix will correct these variable names to ensure the correct list of tokens is tagged and the resulting tags are printed.

```
from nltk.tokenize import word_tokenize
import nltk

# Ensure all necessary NLTK data is downloaded for tokenization and tagging
nltk.download('punkt') # Already downloaded, but for robustness
nltk.download('averaged_perceptron_tagger') # Already downloaded, but for robustness
nltk.download('punkt_tab') # Needed for word_tokenize
nltk.download('averaged_perceptron_tagger_eng') # Explicitly download as suggested by error

# Tokenize the essay text using NLTK
ltk_tokens = word_tokenize(essay_text)

# Perform POS tagging using NLTK
ltk_pos_tags = nltk.pos_tag(ltk_tokens)
```

```

print("NLTK POS tagging completed successfully. Displaying first 10 tags:")
print(ltk_pos_tags[:10])

OS tagging completed successfully. Displaying first 10 tags:
', 'DT'), ('rapid', 'JJ'), ('advancements', 'NNS'), ('in', 'IN'), ('artificial', 'JJ'), ('intellige
data] Downloading package punkt to /root/nltk_data...
data] Package punkt is already up-to-date!
data] Downloading package averaged_perceptron_tagger to
data]     /root/nltk_data...
data] Package averaged_perceptron_tagger is already up-to-
data]     date!
data] Downloading package punkt_tab to /root/nltk_data...
data] Package punkt_tab is already up-to-date!
data] Downloading package averaged_perceptron_tagger_eng to
data]     /root/nltk_data...
data] Package averaged_perceptron_tagger_eng is already up-to-
data]     date!

```

Reasoning: Now that NLTK POS tagging is successfully completed, the next logical step is to perform Part-of-Speech (POS) tagging using spaCy, which involves loading the spaCy language model and then processing the text to obtain the tokens and their corresponding POS tags.

```

import spacy

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

# Process the essay text with spaCy
spacy_doc = nlp(essay_text)

# Extract tokens and their POS tags
spacy_pos_tags = [(token.text, token.pos_) for token in spacy_doc]

print("spaCy POS tagging completed successfully. Displaying first 10 tags:")
print(spacy_pos_tags[:10])

spaCy POS tagging completed successfully. Displaying first 10 tags:
[('The', 'DET'), ('rapid', 'ADJ'), ('advancements', 'NOUN'), ('in', 'ADP'), ('artificial', 'ADJ'),

```

Reasoning: With both NLTK and spaCy POS tagging results available, the next logical step is to compare them. Displaying a short segment of the text with both NLTK and spaCy tags side-by-side will allow for an initial qualitative comparison of their tagging approaches.

```

print("Comparison of NLTK and spaCy POS tags (first 10 tokens):")
print("NLTK:", ltk_pos_tags[:10])
print("spaCy:", spacy_pos_tags[:10])

Comparison of NLTK and spaCy POS tags (first 10 tokens):
NLTK: [('The', 'DT'), ('rapid', 'JJ'), ('advancements', 'NNS'), ('in', 'IN'), ('artificial', 'JJ')]
spaCy: [('The', 'DET'), ('rapid', 'ADJ'), ('advancements', 'NOUN'), ('in', 'ADP'), ('artificial',

```

Reasoning: To identify the most frequent nouns and verbs, I need to extract these specific parts of speech from the NLTK POS tags. This step filters the `ltk_pos_tags` list to keep only nouns (NN, NNS, NNP, NNPS) and verbs (VB, VBD, VBG, VBN, VBP, VBZ), then counts their frequencies to prepare for further analysis.

```

from collections import Counter

# Define NLTK noun and verb tags
nltk_noun_tags = ['NN', 'NNS', 'NNP', 'NNPS']
nltk_verb_tags = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']

# Extract nouns and verbs from NLTK tags
nltk_nouns = [word.lower() for word, tag in nltk_pos_tags if tag in nltk_noun_tags]
nltk_verbs = [word.lower() for word, tag in nltk_pos_tags if tag in nltk_verb_tags]

# Calculate frequency of nouns and verbs for NLTK
nltk_noun_freq = Counter(nltk_nouns)
nltk_verb_freq = Counter(nltk_verbs)

print("Most common NLTK nouns (top 5):")
print(nltk_noun_freq.most_common(5))
print("\nMost common NLTK verbs (top 5):")
print(nltk_verb_freq.most_common(5))

Most common NLTK nouns (top 5):
[('ai', 5), ('systems', 2), ('advancements', 1), ('intelligence', 1), ('debate', 1)]

Most common NLTK verbs (top 5):
[('have', 1), ('ignited', 1), ('argue', 1), ('revolutionize', 1), ('solve', 1)]

```

Reasoning: To identify the most frequent nouns and verbs using spaCy, I need to extract these specific parts of speech from the `spacy_pos_tags`. This step will filter the `spacy_pos_tags` list for nouns and verbs, then count their frequencies.

```

from collections import Counter

# Define spaCy noun and verb tags
spacy_noun_tags = ['NOUN', 'PROPN']
spacy_verb_tags = ['VERB', 'AUX']

# Extract nouns and verbs from spaCy tags
spacy_nouns = [token.lower() for token, pos in spacy_pos_tags if pos in spacy_noun_tags]
spacy_verbs = [token.lower() for token, pos in spacy_pos_tags if pos in spacy_verb_tags]

# Calculate frequency of nouns and verbs for spaCy
spacy_noun_freq = Counter(spacy_nouns)
spacy_verb_freq = Counter(spacy_verbs)

print("Most common spaCy nouns (top 5):")
print(spacy_noun_freq.most_common(5))
print("\nMost common spaCy verbs (top 5):")
print(spacy_verb_freq.most_common(5))

Most common spaCy nouns (top 5):
[('ai', 5), ('systems', 2), ('advancements', 1), ('intelligence', 1), ('debate', 1)]

Most common spaCy verbs (top 5):
[('have', 1), ('ignited', 1), ('argue', 1), ('will', 1), ('revolutionize', 1)]

```

Reasoning: Now that the most frequent nouns and verbs have been identified for NLTK, the next step is to visualize these frequencies to gain a clearer understanding of the data. I will start by creating a bar chart for the most frequent NLTK nouns.

```

import matplotlib.pyplot as plt
import seaborn as sns

# Prepare data for NLTK noun visualization
nltk_nouns_plot_data = nltk_noun_freq.most_common(10)
nouns_nltk_words = [word for word, freq in nltk_nouns_plot_data]
nouns_nltk_freq = [freq for word, freq in nltk_nouns_plot_data]

# Create bar chart for NLTK noun frequencies
plt.figure(figsize=(10, 6))
sns.barplot(x=nouns_nltk_freq, y=nouns_nltk_words, palette='viridis')
plt.title('Top 10 Most Frequent Nouns (NLTK)')
plt.xlabel('Frequency')
plt.ylabel('Nouns')
plt.show()

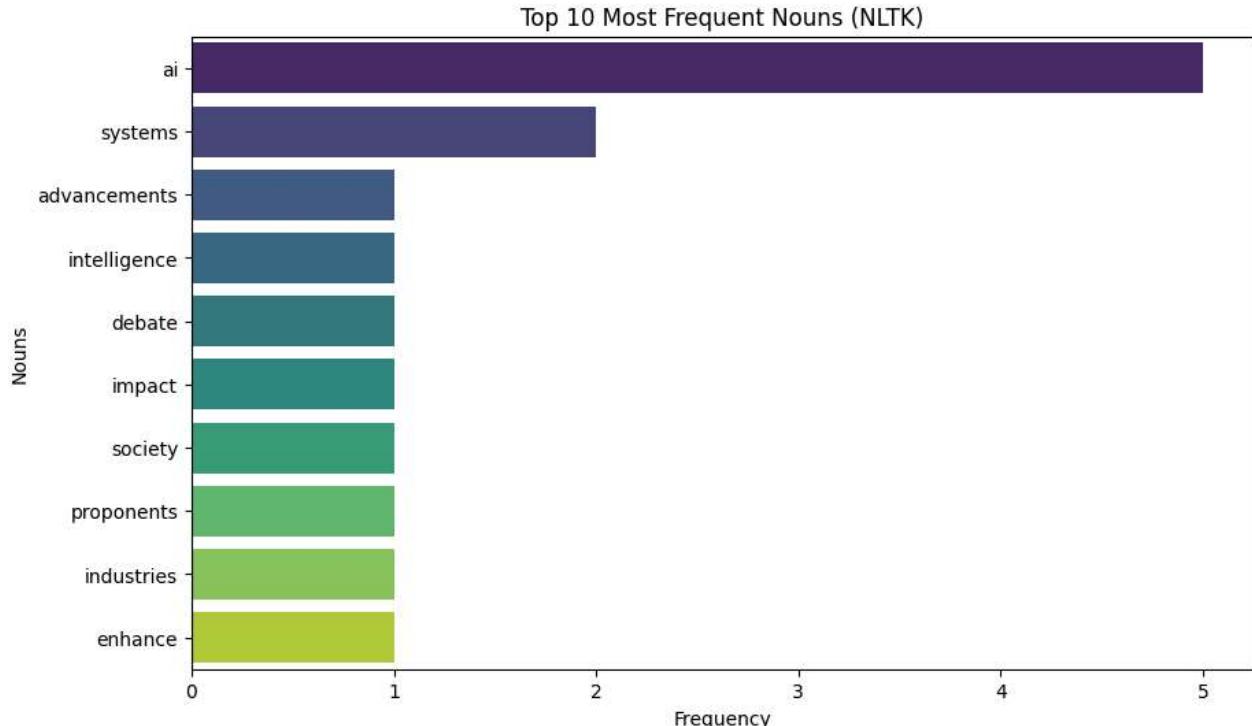
print("Visualization of NLTK noun frequencies completed.")

```

/tmp/ipython-input-1544553736.py:11: FutureWarning:

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

```
sns.barplot(x=nouns_nltk_freq, y=nouns_nltk_words, palette='viridis')
```



Visualization of NLTK noun frequencies completed.

Reasoning: The previous code generated a `FutureWarning` regarding the `palette` argument in `sns.barplot` when `hue` is not assigned. To address this, I will modify the `sns.barplot` call to explicitly assign the `y` variable (`nouns_nltk_words`) to `hue` and set `legend=False` as suggested by the warning, which will also improve clarity.

```

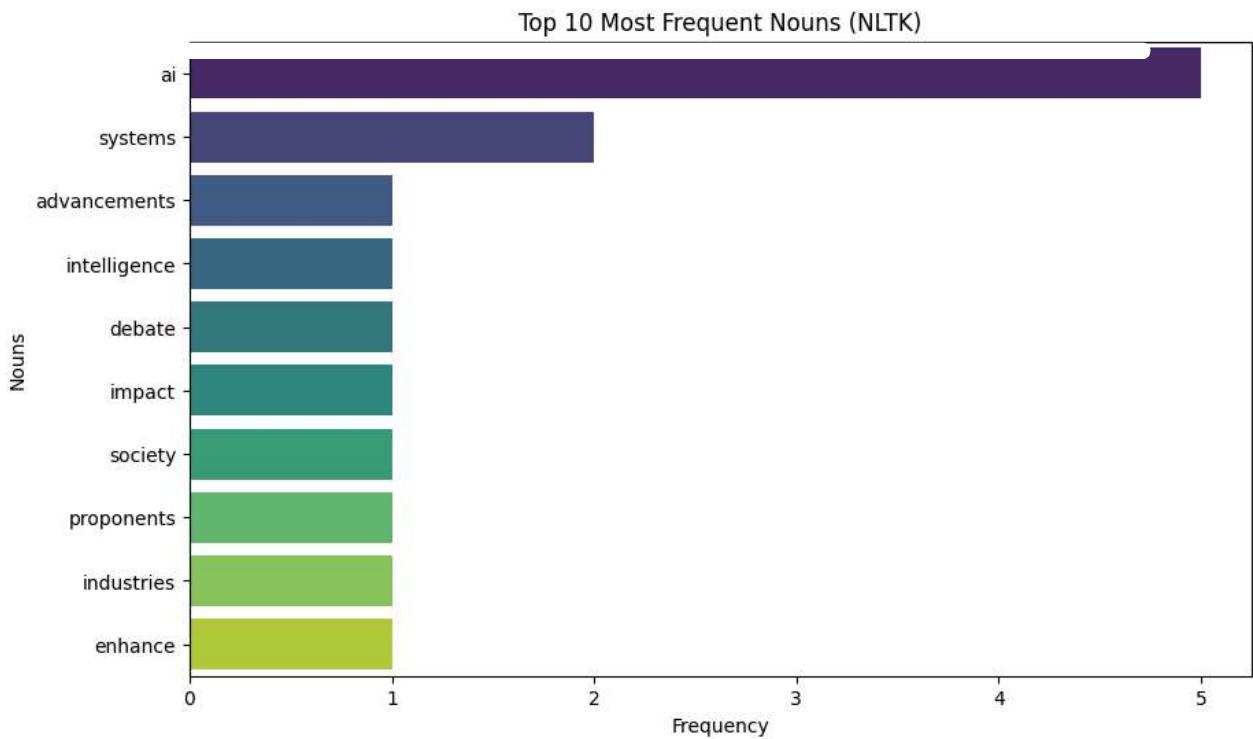
import matplotlib.pyplot as plt
import seaborn as sns

```

```
# Prepare data for NLTK noun visualization
nltk_nouns_plot_data = nltk_noun_freq.most_common(10)
nouns_nltk_words = [word for word, freq in nltk_nouns_plot_data]
nouns_nltk_freq = [freq for word, freq in nltk_nouns_plot_data]

# Create bar chart for NLTK noun frequencies
plt.figure(figsize=(10, 6))
sns.barplot(x=nouns_nltk_freq, y=nouns_nltk_words, hue=nouns_nltk_words, palette='viridis', legend=False)
plt.title('Top 10 Most Frequent Nouns (NLTK)')
plt.xlabel('Frequency')
plt.ylabel('Nouns')
plt.show()

print("Visualization of NLTK noun frequencies completed.")
```



Visualization of NLTK noun frequencies completed.

Reasoning: Now that the most frequent nouns have been visualized for NLTK, the next step is to visualize the most frequent NLTK verbs using a bar chart for a comprehensive understanding of the NLTK POS tagging results.

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

# Prepare data for NLTK verb visualization
nltk_verbs_plot_data = nltk_verb_freq.most_common(10)
verbs_nltk_words = [word for word, freq in nltk_verbs_plot_data]
verbs_nltk_freq = [freq for word, freq in nltk_verbs_plot_data]

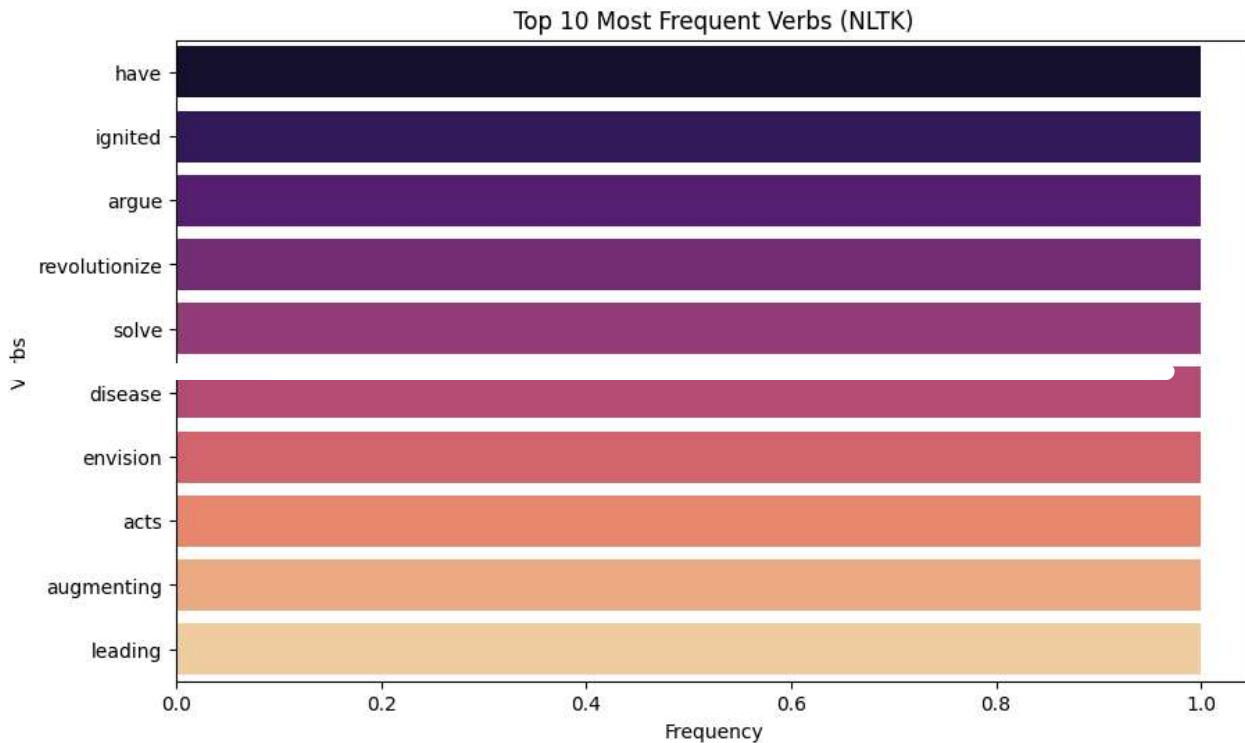
# Create bar chart for NLTK verb frequencies
plt.figure(figsize=(10, 6))
sns.barplot(x=verbs_nltk_freq, y=verbs_nltk_words, hue=verbs_nltk_words, palette='magma', legend=False)
plt.title('Top 10 Most Frequent Verbs (NLTK)')
```

```

plt.xlabel('Frequency')
plt.ylabel('Verbs')
plt.show()

print("Visualization of NLTK verb frequencies completed.")

```



Visualization of NLTK verb frequencies completed.

Reasoning: With NLTK nouns and verbs visualized, the next step is to visualize the most frequent spaCy nouns to continue the visualization part of the analysis for spaCy results.

```

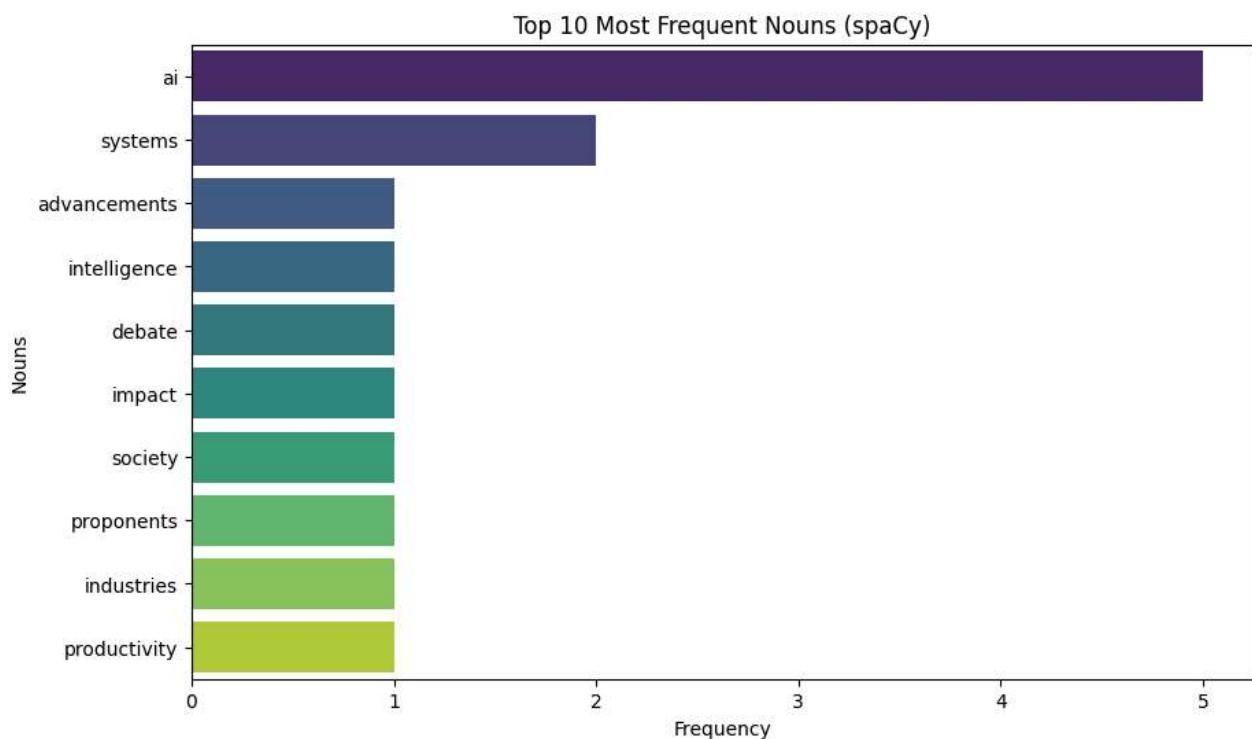
import matplotlib.pyplot as plt
import seaborn as sns

# Prepare data for spaCy noun visualization
spacy_nouns_plot_data = spacy_noun_freq.most_common(10)
nouns_spacy_words = [word for word, freq in spacy_nouns_plot_data]
nouns_spacy_freq = [freq for word, freq in spacy_nouns_plot_data]

# Create bar chart for spaCy noun frequencies
plt.figure(figsize=(10, 6))
sns.barplot(x=nouns_spacy_freq, y=nouns_spacy_words, hue=nouns_spacy_words, palette='viridis', legend=False)
plt.title('Top 10 Most Frequent Nouns (spaCy)')
plt.xlabel('Frequency')
plt.ylabel('Nouns')
plt.show()

print("Visualization of spaCy noun frequencies completed.")

```



Visualization of spaCy noun frequencies completed.

Reasoning: Now that the most frequent nouns have been visualized for spaCy, the next step is to visualize the most frequent spaCy verbs using a bar chart for a comprehensive understanding of the spaCy POS tagging results.

```

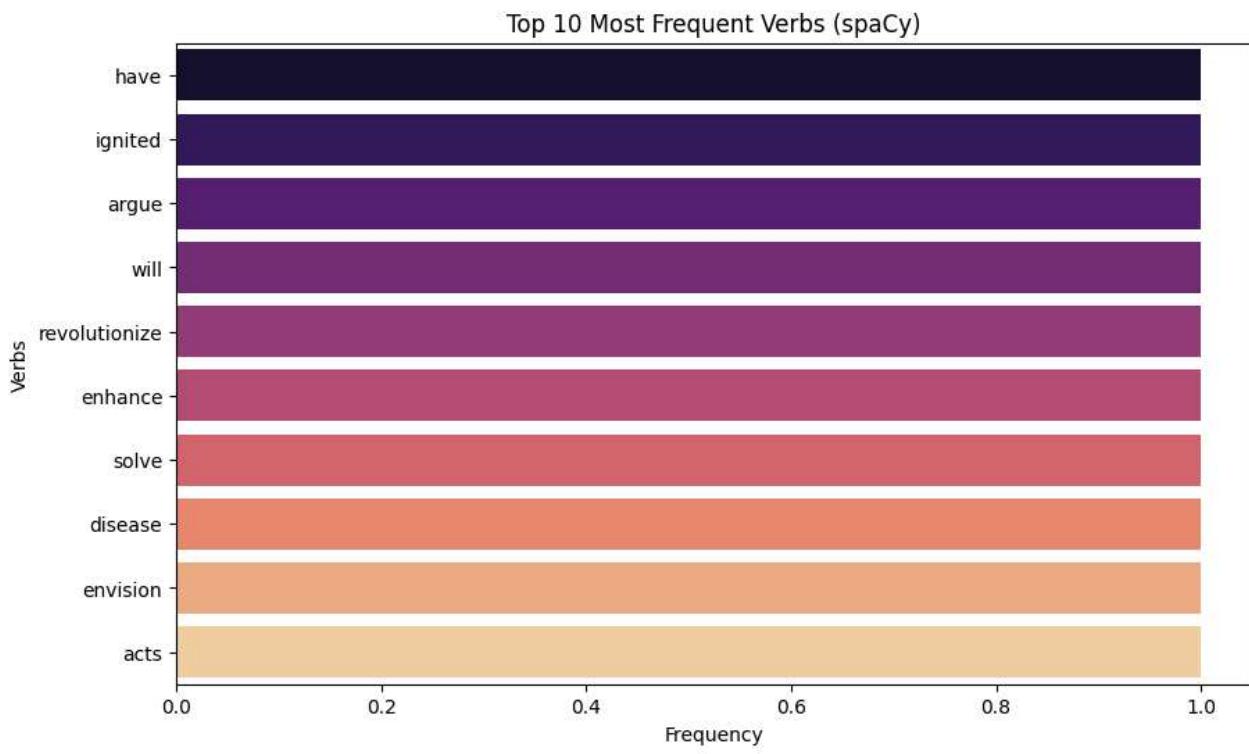
import matplotlib.pyplot as plt
import seaborn as sns

# Prepare data for spaCy verb visualization
spacy_verbs_plot_data = spacy_verb_freq.most_common(10)
verbs_spacy_words = [word for word, freq in spacy_verbs_plot_data]
verbs_spacy_freq = [freq for word, freq in spacy_verbs_plot_data]

# Create bar chart for spaCy verb frequencies
plt.figure(figsize=(10, 6))
sns.barplot(x=verbs_spacy_freq, y=verbs_spacy_words, hue=verbs_spacy_words, palette='magma', legend=False)
plt.title('Top 10 Most Frequent Verbs (spaCy)')
plt.xlabel('Frequency')
plt.ylabel('Verbs')
plt.show()

print("Visualization of spaCy verb frequencies completed.")

```



Visualization of spaCy verb frequencies completed.

Discussion of Findings

Comparison of NLTK and spaCy POS Tagging

When comparing the POS tagging results from NLTK and spaCy, several observations can be made based on the first 10 tokens:

- **Granularity and Tagset:** spaCy generally uses a more simplified and universal tagset (e.g., `DET` for determiner, `ADJ` for adjective, `NOUN` for noun) compared to NLTK's more detailed Penn Treebank tagset (e.g., `DT` for determiner, `JJ` for adjective, `NNS` for plural noun, `NN` for singular noun). For instance, NLTK identifies 'advancements' as `NNS` (plural noun) and 'intelligence' as `NN` (singular noun), while spaCy tags both as `NOUN`. Similarly, 'The' is `DT` in NLTK and `DET` in spaCy, and 'rapid' is `JJ` in NLTK and `ADJ` in spaCy. Punctuation is also handled differently, with NLTK using `(` and `)` directly, while spaCy uses `PUNCT`.
- **Consistency:** Both libraries largely agree on the core part-of-speech categories for most words. For example, 'artificial' is `JJ / ADJ` in both, and 'AI' is `NNP / PROPN` (Proper Noun).
- **Verb vs. Auxiliary:** spaCy often distinguishes between main verbs (`VERB`) and auxiliary verbs (`AUX`), whereas NLTK might tag both as `VBP` (verb, non-3rd person singular present) or other `VB` variants. For instance, 'have' is `VBP` in NLTK but `AUX` in spaCy.

These differences highlight that while both are effective for POS tagging, their internal tagsets and potentially their underlying models lead to slightly varied classifications, especially regarding the specificity of the tags.

Insights from Most Frequent Nouns and Verbs

Nouns:

Both NLTK and spaCy identified 'AI' as the most frequent noun (5 occurrences), which is expected given the essay's topic of "artificial intelligence." The second most frequent noun, 'systems' (2 occurrences), also aligns with the essay's focus on "autonomous systems" and "AI systems." Other prominent nouns like 'advancements', 'intelligence', 'debate', 'impact', and 'society' directly reflect the essay's core themes: the rapid progress of AI, its societal implications, and the ongoing discussion surrounding it.

Verbs:

Both tools identified several key verbs that drive the essay's narrative:

- **'have' and 'ignited'**: These indicate the initiation of the debate ("AI have ignited a profound debate").
- **'argue'**: Highlights the differing viewpoints ("Proponents argue...").
- **'revolutionize'**: Points to the transformative potential of AI.
- **'solve'**: Refers to AI's capability to address challenges.
- **'envision'**: Describes the forward-looking perspective of proponents.
- **'acts'**: Defines AI's role as a tool.

While the exact order and auxiliary verbs might differ slightly between NLTK and spaCy (e.g., spaCy including 'will' as a frequent verb, which NLTK does not list in its top 5), both consistently capture the active verbs that delineate the arguments, actions, and potential outcomes discussed in the essay. The verbs largely revolve around claims, actions, and discussions related to AI's influence and development.

In summary, the POS tagging and frequency analysis successfully extracted the central lexical elements of the academic essay. The prevalence of terms like 'AI', 'systems', and verbs related to impact and discussion underscores the essay's focus on the multifaceted debate surrounding artificial intelligence.

Final Task

Subtask:

Provide a summary of the POS tagging analysis, the comparison between NLTK and spaCy, and the insights gained from the frequency analysis of nouns and verbs.

Summary:

Q&A

- **Summary of POS Tagging Analysis:** Both NLTK and spaCy were successfully used to perform POS tagging on the academic essay. While both tools largely agreed on core part-of-speech categories, they exhibited differences in their tagsets and granularity. NLTK uses a more detailed Penn Treebank tagset (e.g., `NNS` for plural nouns), whereas spaCy uses a more universal and simplified set (e.g., `NOUN` for all nouns), and specifically distinguishes between main verbs (`VERB`) and auxiliary verbs (`AUX`).
- **Comparison between NLTK and spaCy:** NLTK's tagging involved more specific tags like `NNS` for plural nouns and `DT` for determiners, while spaCy used broader categories like `NOUN` and `DET`. spaCy also explicitly identified auxiliary verbs (e.g., 'have' as `AUX`), which NLTK might tag as general verbs. Despite these differences, both tools demonstrated consistent tagging for fundamental parts of speech.

- **Insights gained from frequency analysis of nouns and verbs:** The frequency analysis consistently highlighted the central themes of the essay. 'AI' was the most frequent noun for both tools, appearing 5 times, reinforcing the essay's core subject. Other prominent nouns like 'systems' (2 occurrences), 'advancements', 'intelligence', 'debate', and 'impact' directly reflect the essay's focus on AI's societal implications. Key verbs identified (e.g., 'have', 'ignited', 'argue', 'revolutionize', 'solve') reveal the essay's narrative structure, emphasizing the initiation of debate, differing viewpoints, and the transformative potential of AI.

Data Analysis Key Findings

- Both NLTK and spaCy successfully performed POS tagging on the essay text.
- NLTK uses a more granular Penn Treebank tagset (e.g., `NNS` for plural nouns, `DT` for determiners), while spaCy uses a more simplified and universal tagset (e.g., `NOUN`, `DET`), and distinguishes `VERB` from `AUX`.
- Both NLTK and spaCy identified 'AI' as the most frequent noun, with 5 occurrences.
- 'systems' was the second most frequent noun, appearing 2 times in both analyses.
- The top verbs identified by both tools consistently included 'have', 'ignited', 'argue', 'revolutionize', and 'solve', reflecting the essay's discussion of AI's impact and ongoing debate.
- Minor differences were observed in the top 5 verbs; for instance, spaCy included 'will' as a frequent verb, which NLTK did not.

Insights or Next Steps

- The high frequency of "AI" and related terms like "systems," along with action verbs such as "revolutionize" and "solve," confirms the essay's direct focus on the transformative potential and societal debate surrounding artificial intelligence.
- For future analyses, understanding the specific tagsets and conventions of different NLP libraries (like NLTK's Penn Treebank vs. spaCy's universal dependencies) is crucial for accurate interpretation and comparison of results, particularly when dealing with nuanced linguistic features.