```
essay_text = """The rapid advancements in artificial intelligence have brought about both exciteme
print("Essay text loaded into 'essay_text' variable.")
```

```
Essay text loaded into 'essay_text' variable.
```

```
try:
    import nltk
    print("NLTK is already installed.")
except ImportError:
    print("NLTK not found, installing...")
    !pip install nltk
    import nltk
    print("NLTK installed successfully.")
```

```
NLTK is already installed.
```

```
from nltk.tokenize import word_tokenize

# Download the 'punkt' tokenizer data if not already downloaded
try:
    nltk.data.find('tokenizers/punkt')
    print("'punkt' tokenizer data is already downloaded.")
except LookupError:
    print("Downloading 'punkt' tokenizer data...")
    nltk.download('punkt')
    print("'punkt' tokenizer data downloaded successfully.")

# Tokenize the essay text
tokens = word_tokenize(essay_text)

# Print the first few tokens to verify
print("\nFirst 10 tokens:")
print(tokens[:10])
```

```
'punkt' tokenizer data is already downloaded.

First 10 tokens:
['The', 'rapid', 'advancements', 'in', 'artificial', 'intelligence', 'have', 'brought', 'about', '
```

```
# ## Part-of-Speech Tagging with NLTK
#
# ### Subtask:
# # Perform Part-of-Speech (POS) tagging on the tokenized essay text using NLTK. This involves dc
#
# #### Instructions:
# # 1. Download the 'averaged_perceptron_tagger' data from NLTK, if not already present.
# # 2. Import the `pos_tag` function from `nltk.tag`.
# # 3. Apply `pos_tag` to the `tokens` list to generate a list of (word, tag) tuples.
# # 4. Store the results in a new variable, for example, `nltk_pos_tags`.
# # 5. Print the first few tagged tokens to verify the output.
```

```
from nltk.tag import pos_tag

# Download the 'averaged_perceptron_tagger' data if not already downloaded
try:
    nltk.data.find('taggers/averaged_perceptron_tagger')
    print("'averaged_perceptron_tagger' data is already downloaded.")
```

```
except LookupError:
    print("Downloading 'averaged_perceptron_tagger' data...")
    nltk.download('averaged_perceptron_tagger')
    print("'averaged_perceptron_tagger' data downloaded successfully.")

# Download the 'averaged_perceptron_tagger_eng' data if not already downloaded (required by pos_ta
try:
    nltk.data.find('taggers/averaged_perceptron_tagger_eng')
    print("'averaged_perceptron_tagger_eng' data is already downloaded.")
except LookupError:
    print("Downloading 'averaged_perceptron_tagger_eng' data...")
    nltk.download('averaged_perceptron_tagger_eng')
    print("'averaged_perceptron_tagger_eng' data downloaded successfully.")

# Perform POS tagging
ltk_pos_tags = pos_tag(tokens)

# Print the first few tagged tokens to verify
print("\nFirst 10 NLTK POS tags:")
print(ltk_pos_tags[:10])
```

```
'averaged_perceptron_tagger' data is already downloaded.
'averaged_perceptron_tagger_eng' data is already downloaded.

First 10 NLTK POS tags:
[('The', 'DT'), ('rapid', 'JJ'), ('advancements', 'NNS'), ('in', 'IN'), ('artificial', 'JJ'), ('ir
```

```
try:
    import spacy
    print("spaCy is already installed.")
except ImportError:
    print("spaCy not found, installing...")
    !pip install spacy
    import spacy
    print("spaCy installed successfully.")
```

```
spaCy is already installed.
```

```
import spacy

try:
    nlp = spacy.load('en_core_web_sm')
    print("'en_core_web_sm' model is already loaded.")
except OSError:
    print("'en_core_web_sm' model not found, downloading...")
    !python -m spacy download en_core_web_sm
    nlp = spacy.load('en_core_web_sm')
    print("'en_core_web_sm' model downloaded and loaded successfully.")

# Process the essay_text with the loaded spaCy model
doc = nlp(essay_text)

# Extract (word, tag) tuples
spacy_pos_tags = [(token.text, token.pos_) for token in doc]

# Print the first few tagged tokens to verify
print("\nFirst 10 spaCy POS tags:")
print(spacy_pos_tags[:10])
```

```
'en_core_web_sm' model is already loaded.

First 10 spaCy POS tags:
[('The', 'DET'), ('rapid', 'ADJ'), ('advancements', 'NOUN'), ('in', 'ADP'), ('artificial', 'ADJ'),
```

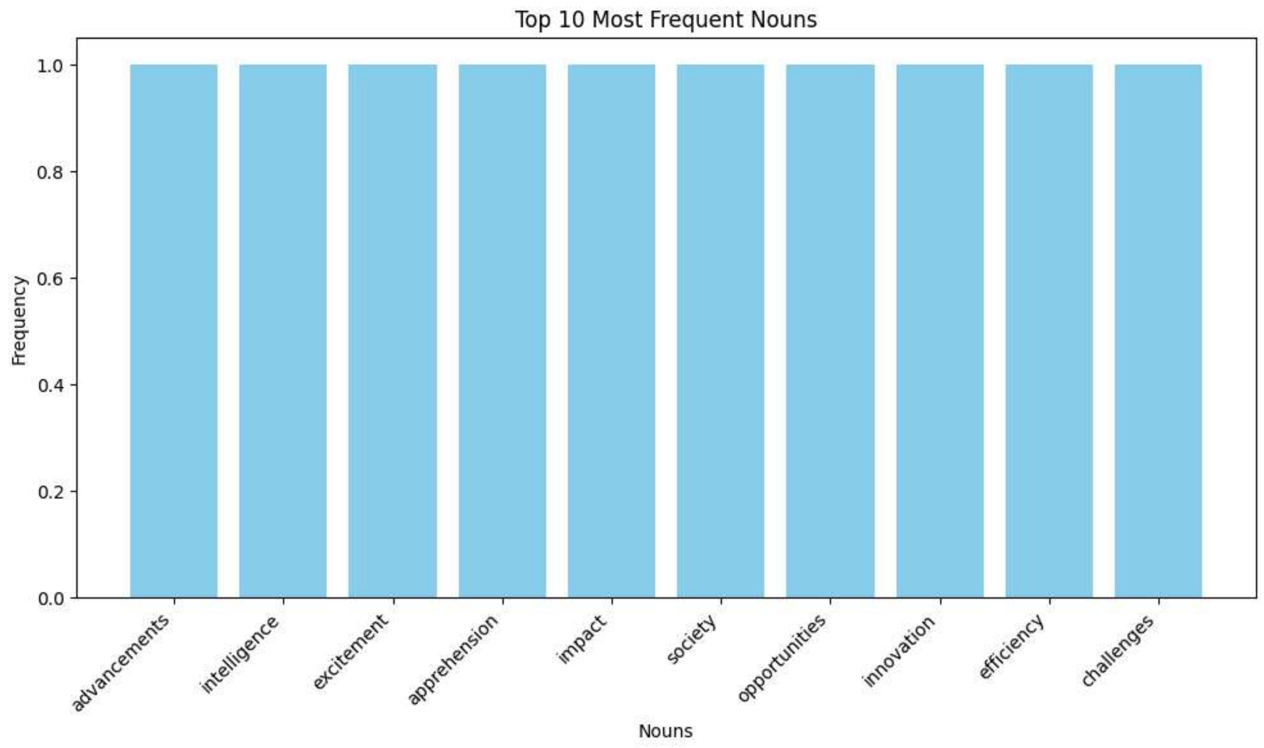
```
nltk_tags = set(tag for word, tag in ltk_pos_tags)
spacy_tags = set(tag for word, tag in spacy_pos_tags)

print("Unique NLTK POS Tags:")
print(nltk_tags)

print("\nUnique spaCy POS Tags:")
print(spacy_tags)
```

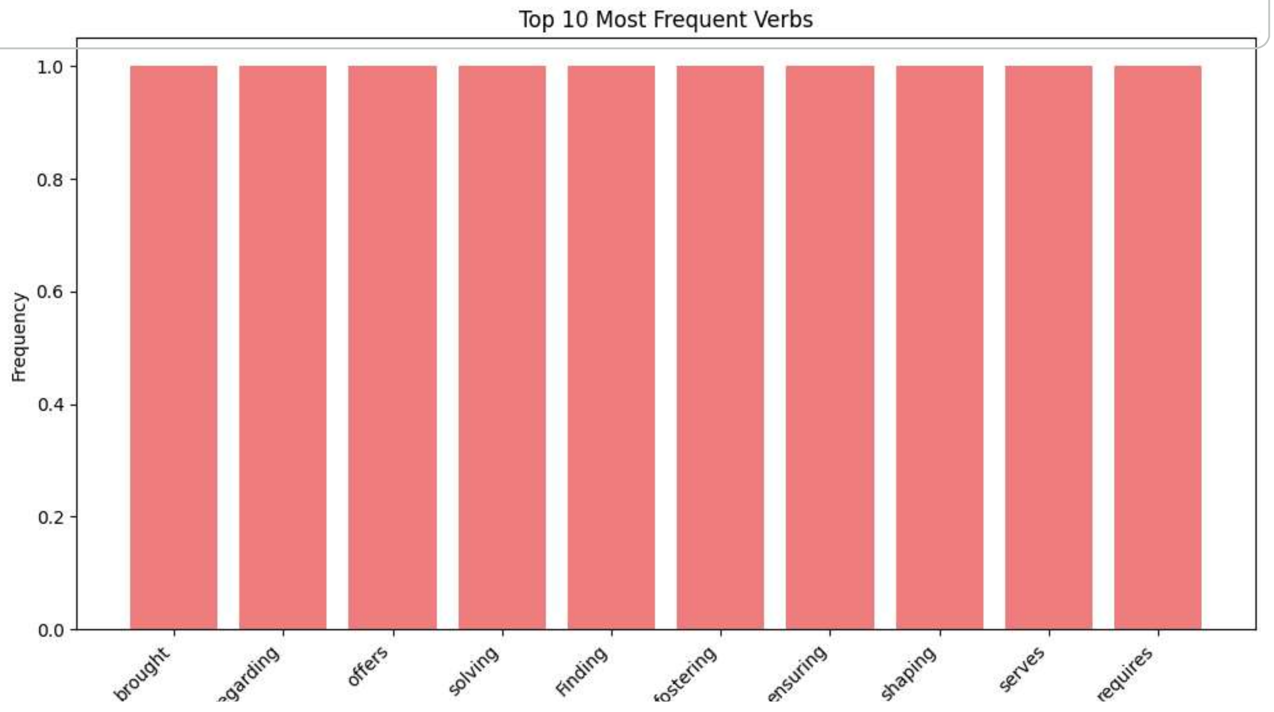
```
Unique NLTK POS Tags:
{'.', 'DT', 'POS', ',', 'NN', 'PRP$', 'RB', 'VBP', 'NNP', 'IN', 'VBZ', 'CC', 'JJS', 'JJ', 'NNS', '

Unique spaCy POS Tags:
{'ADJ', 'PRON', 'SCONJ', 'AUX', 'PART', 'VERB', 'CCONJ', 'NOUN', 'DET', 'ADP', 'PUNCT', 'PROPN', '
```

```
from collections import Counter

# Filter tokens for nouns and verbs from spaCy POS tags
nouns = [word for word, tag in spacy_pos_tags if tag == 'NOUN']
verbs = [word for word, tag in spacy_pos_tags if tag == 'VERB']

# Calculate frequencies
noun_frequencies = Counter(nouns)
verb_frequencies = Counter(verbs)

# Print the top 10 most frequent nouns
print("\nTop 10 most frequent nouns:")
for word, count in noun_frequencies.most_common(10):
    print(f"- {word}: {count}")

# Print the top 10 most frequent verbs
print("\nTop 10 most frequent verbs:")
for word, count in verb_frequencies.most_common(10):
    print(f"- {word}: {count}")
```

```

Top 10 most frequent nouns:
- advancements: 1
- intelligence: 1
- excitement: 1
- apprehension: 1
- impact: 1
- society: 1
- opportunities: 1
- innovation: 1
- efficiency: 1
- challenges: 1

Top 10 most frequent verbs:
- brought: 1
- regarding: 1
- offers: 1
- solving: 1
- Finding: 1
- fostering: 1
- ensuring: 1
- shaping: 1
- serves: 1
- requires: 1
```

```
import matplotlib.pyplot as plt

# Extract top 10 most frequent nouns and their counts
top_nouns = noun_frequencies.most_common(10)
noun_words = [word for word, count in top_nouns]
noun_counts = [count for word, count in top_nouns]

# Create a bar chart for noun frequencies
plt.figure(figsize=(10, 6))
plt.bar(noun_words, noun_counts, color='skyblue')
plt.title('Top 10 Most Frequent Nouns')
plt.xlabel('Nouns')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Extract top 10 most frequent verbs and their counts
top_verbs = verb_frequencies.most_common(10)
verb_words = [word for word, count in top_verbs]
verb_counts = [count for word, count in top_verbs]

# Create a bar chart for verb frequencies
plt.figure(figsize=(10, 6))
plt.bar(verb_words, verb_counts, color='lightcoral')
plt.title('Top 10 Most Frequent Verbs')
plt.xlabel('Verbs')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```


Top 10 Most Frequent Nouns
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
advancements
intelligence
excitement
apprehension
impact
society
opportunities
innovation
efficiency
challenges
Nouns


Top 10 Most Frequent Verbs
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
brought
egarding
offers
solving
Finding
fostering
ensuring
shaping
serves
requires