

## Notebook Gemini



Start coding or generate with AI.

```
pip install nltk spacy
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages (3.8.11)
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) (2025.11.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.15)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.13)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (8.3.10)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srslry<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.5.2)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.4.3)
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.21.1)
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) (2.32.4)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.1)
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!=1.8,!=1.8.1,<3.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.1)
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.1)
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.1)
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.1)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.0.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.1.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.1.0)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.3.0)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (0.0.1)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (0.7.0)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (5.2.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->spacy) (3.0.3)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.8.11)
```

```
import nltk
import spacy

# Download NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
print("NLTK punkt and averaged_perceptron_tagger downloaded.")

# Download spaCy English model
# Using 'en_core_web_sm' for a small model
try:
    spacy.load('en_core_web_sm')
except OSError:
    print('Downloading spaCy model en_core_web_sm...')
    !python -m spacy download en_core_web_sm
    print('SpaCy model en_core_web_sm downloaded and loaded.')

# Define the academic essay text
essay_text = """The impact of artificial intelligence (AI) on modern society is profound and multifaceted, necessitating a comprehensive understanding of its ethical implications and societal impacts. As AI continues to advance, it is crucial to address the potential risks and benefits of this technology, particularly in areas such as privacy, bias, and accountability. This requires a multidisciplinary approach that considers the social, cultural, and political contexts in which AI is developed and deployed. By fostering a responsible and ethical development of AI, we can ensure that it serves the greater good and promotes a more just and equitable society. In conclusion, the impact of AI on modern society is significant and far-reaching, requiring a concerted effort to harness its potential while mitigating its risks. As we move forward, it is essential to prioritize the well-being of all individuals and communities in the face of this rapidly changing technological landscape."""
print("Academic essay text loaded.")

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]      /root/nltk_data...
[nltk_data]  Unzipping taggers/averaged_perceptron_tagger.zip.
NLTK punkt and averaged_perceptron_tagger downloaded.
Academic essay text loaded.
```

```

import nltk
import spacy

# Download NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
print("NLTK punkt and averaged_perceptron_tagger downloaded.")

# Download and load spaCy English model
try:
    nlp = spacy.load('en_core_web_sm')
    print('SpaCy model en_core_web_sm loaded.')
except OSError:
    print('Downloading spaCy model en_core_web_sm...')
    !python -m spacy download en_core_web_sm
    nlp = spacy.load('en_core_web_sm') # Load after download
    print('SpaCy model en_core_web_sm downloaded and loaded.')

# Define the academic essay text
essay_text = """The impact of artificial intelligence (AI) on modern society is profound and multifaceted, necessitating a comp

print("Academic essay text loaded.")

[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 punkt and averaged_perceptron_tagger downloaded.
SpaCy model en_core_web_sm loaded.
Academic essay text loaded.

```

```

import nltk

# Ensure all necessary NLTK data is downloaded for tokenization and POS tagging
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')
print("All required NLTK data downloaded.")

# 1. Tokenize the essay_text into individual words using nltk.word_tokenize()
nltk_tokens = nltk.word_tokenize(essay_text)

# 2. Perform Part-of-Speech (POS) tagging on the nltk_tokens using nltk.pos_tag()
nltk_pos_tags = nltk.pos_tag(nltk_tokens);

# 3. Print the first few tokens and their corresponding POS tags
print("\nFirst 10 NLTK tokens and their POS tags:")
for word, tag in nltk_pos_tags[:10]:
    print(f"Word: {word}, Tag: {tag}")

print(f"\nTotal NLTK tokens: {len(nltk_tokens)}")
print(f"Total NLTK POS tags: {len(nltk_pos_tags)}")

[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]   Package punkt_tab 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 averaged_perceptron_tagger_eng to
[nltk_data]   /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
All required NLTK data downloaded.

First 10 NLTK tokens and their POS tags:
Word: The, Tag: DT
Word: impact, Tag: NN
Word: of, Tag: IN
Word: artificial, Tag: JJ
Word: intelligence, Tag: NN
Word: (, Tag: (
Word: AI, Tag: NNP
Word: ), Tag: )
Word: on, Tag: IN

```

```
Word: modern, Tag: JJ
```

```
Total NLTK tokens: 248
```

```
Total NLTK POS tags: 248
```

```
import spacy

# 1. Process the essay_text using the loaded spaCy model
doc = nlp(essay_text)

# 2. Iterate through the first 10 tokens and print text and universal POS tag
print("First 10 spaCy tokens and their universal POS tags:")
for i, token in enumerate(doc):
    if i >= 10:
        break
    print(f"Word: {token.text}, Tag: {token.pos_}")

# 3. Print the total number of tokens processed by spaCy
print(f"\nTotal spaCy tokens: {len(doc)}")
```

```
First 10 spaCy tokens and their universal POS tags:
```

```
Word: The, Tag: DET
Word: impact, Tag: NOUN
Word: of, Tag: ADP
Word: artificial, Tag: ADJ
Word: intelligence, Tag: NOUN
Word: (, Tag: PUNCT
Word: AI, Tag: PROPN
Word: ), Tag: PUNCT
Word: on, Tag: ADP
Word: modern, Tag: ADJ
```

```
Total spaCy tokens: 252
```

```
print("\n--- Comparing NLTK and spaCy POS Tags ---\n")

# Choose a representative sample size for comparison (e.g., the first 20 tokens)
sample_size = 20

print(f"Comparison of the first {sample_size} tokens:")
print("-----")
print(f"{'Word':<15} {'NLTK Tag':<15} {'spaCy Tag':<15}")
print("-----")

for i in range(min(sample_size, len(nltk_pos_tags), len(doc))):
    nltk_word, nltk_tag = nltk_pos_tags[i]
    spacy_token = doc[i]
    spacy_word = spacy_token.text
    spacy_tag = spacy_token.pos_

    # Align output, assuming words might slightly differ due to tokenization differences
    # For simplicity, we'll use the NLTK word if they match, or print both if they differ significantly
    display_word = nltk_word if nltk_word == spacy_word else f"{nltk_word}/{spacy_word}"

    print(f"{display_word:<15} {nltk_tag:<15} {spacy_tag:<15}")

print("-----")
```

```
--- Comparing NLTK and spaCy POS Tags ---
```

```
Comparison of the first 20 tokens:
```

Word	NLTK Tag	spaCy Tag
The	DT	DET
impact	NN	NOUN
of	IN	ADP
artificial	JJ	ADJ
intelligence	NN	NOUN
(	(	PUNCT
AI	NNP	PROPN
)	)	PUNCT
on	IN	ADP
modern	JJ	ADJ
society	NN	NOUN
is	VBZ	AUX
profound	JJ	ADJ
and	CC	CCONJ

multifaceted	VBD	VERB
,	,	PUNCT
necessitating	VBG	VERB
a	DT	DET
comprehensive	JJ	ADJ
analysis	NN	NOUN
-----		

\*\*Observation on NLTK vs. spaCy POS Tags:\*\*

From the sample comparison, we can observe key differences and similarities:

- \* \*\*Granularity vs. Universality:\*\* NLTK's tag set (e.g., 'DT' for determiner, 'NN' for noun, 'NNP' for proper noun, 'VBZ' for past tense verb) provides more granular information.
- \* \*\*spaCy's Universal Tag Set:\*\* spaCy, on the other hand, uses a more universal and simplified tag set ('DET' for determiner, 'NOUN' for noun, 'VERB' for verb, etc.).
- \* \*\*Consistency for common types:\*\* For common word types like adjectives ('JJ' in NLTK, 'ADJ' in spaCy) and conjunctions ('CC' in NLTK, 'CONJ' in spaCy), the tags are consistent.
- \* \*\*Punctuation:\*\* Both libraries generally agree on punctuation tags (`'` and `)` in NLTK, 'PUNCT' in spaCy for parentheses; `)` in spaCy).
- \* \*\*Tokenization Differences:\*\* Minor differences in tokenization might lead to slight mismatches in word counts or token alignment.

In summary, NLTK provides a more detailed, Penn Treebank-style tag set, while spaCy offers a broader, universal tag set, which can be more useful for cross-language processing.

```
academic_concepts = []
arguments = []

# Iterate through each token in the spaCy doc object
for token in doc:
    # Check if the token is a noun (NOUN or PROPN)
    if token.pos_ in ['NOUN', 'PROPN']:
        academic_concepts.append(token.lemma_.lower())
    # Check if the token is a verb (VERB or AUX)
    elif token.pos_ in ['VERB', 'AUX']:
        arguments.append(token.lemma_.lower())

print("First 10 extracted academic concepts (nouns):")
print(academic_concepts[:10])

print("\nFirst 10 extracted arguments (verbs):")
print(arguments[:10])

print(f"\nTotal academic concepts extracted: {len(academic_concepts)}")
print(f"Total arguments extracted: {len(arguments)}")

First 10 extracted academic concepts (nouns):
['impact', 'intelligence', 'ai', 'society', 'analysis', 'implication', 'ai', 'technology', 'machine', 'learning']

First 10 extracted arguments (verbs):
['be', 'multifacete', 'necessitate', 'range', 'be', 'integrate', 'include', 'argue', 'offer', 'raise']

Total academic concepts extracted: 82
Total arguments extracted: 39
```

```
from collections import Counter
import pandas as pd

# 1. Calculate the frequency of each academic concept
concept_frequencies = Counter(academic_concepts)

# 2. Create a pandas DataFrame for academic concepts
df_concepts = pd.DataFrame(concept_frequencies.items(), columns=['Concept', 'Frequency'])
df_concepts = df_concepts.sort_values(by='Frequency', ascending=False).reset_index(drop=True)

# 3. Calculate the frequency of each argument
argument_frequencies = Counter(arguments)

# 4. Create a pandas DataFrame for arguments
df_arguments = pd.DataFrame(argument_frequencies.items(), columns=['Argument', 'Frequency'])
df_arguments = df_arguments.sort_values(by='Frequency', ascending=False).reset_index(drop=True)

# 5. Print the top 10 academic concepts and their frequencies
print("\nTop 10 Academic Concepts and their Frequencies:")
print(df_concepts.head(10))

# 6. Print the top 10 arguments and their frequencies
```

```
print("\nTop 10 Arguments and their Frequencies:")
print(df_arguments.head(10))
```

Top 10 Academic Concepts and their Frequencies:

	Concept	Frequency
0	ai	10
1	risk	2
2	deployment	2
3	education	2
4	learning	2
5	analysis	1
6	society	1
7	intelligence	1
8	impact	1
9	algorithm	1

Top 10 Arguments and their Frequencies:

	Argument	Frequency
0	be	8
1	argue	2
2	can	2
3	range	1
4	necessitate	1
5	multifacete	1
6	include	1
7	integrate	1
8	offer	1
9	raise	1

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

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

# 1. Create a bar chart for the top 10 academic concepts
plt.figure(figsize=(12, 6))
sns.barplot(x='Concept', y='Frequency', data=df_concepts.head(10), palette='viridis')
plt.title('Top 10 Academic Concepts by Frequency', fontsize=16)
plt.xlabel('Academic Concept', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()

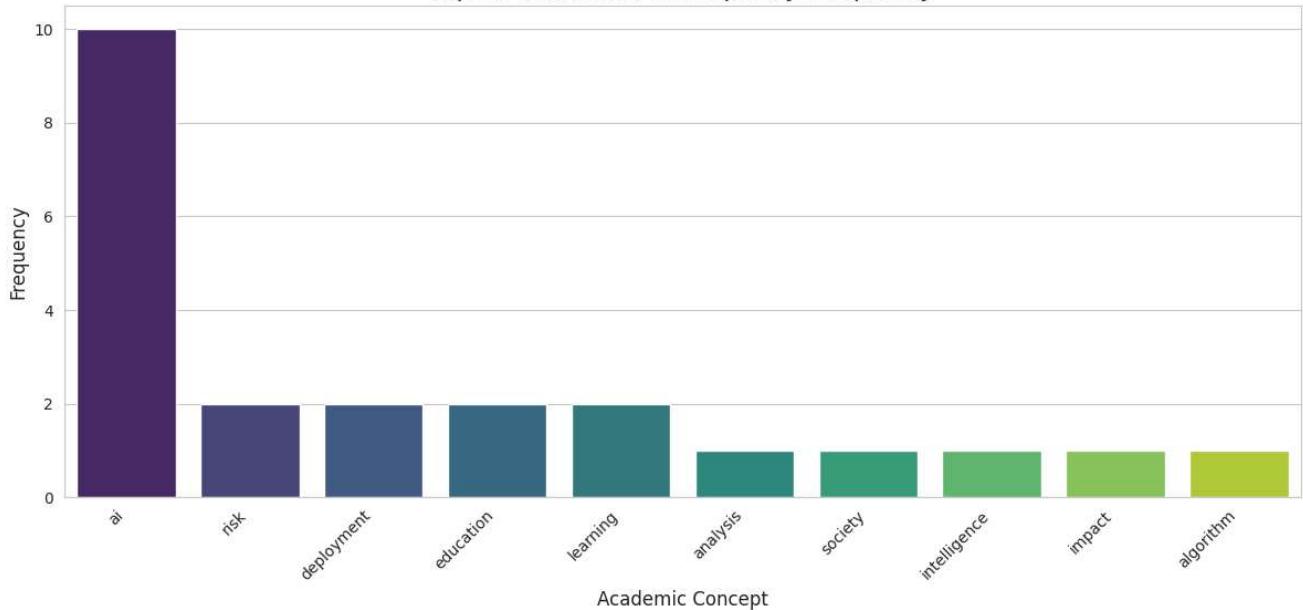
# 2. Create a bar chart for the top 10 arguments (verbs)
plt.figure(figsize=(12, 6))
sns.barplot(x='Argument', y='Frequency', data=df_arguments.head(10), palette='plasma')
plt.title('Top 10 Arguments (Verbs) by Frequency', fontsize=16)
plt.xlabel('Argument', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2564200704.py:9: FutureWarning:
```

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

```
sns.barplot(x='Concept', y='Frequency', data=df_concepts.head(10), palette='viridis')
```

Top 10 Academic Concepts by Frequency



```
/tmp/ipython-input-2564200704.py:20: FutureWarning:
```

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

```
sns.barplot(x='Argument', y='Frequency', data=df_arguments.head(10), palette='plasma')
```

Top 10 Arguments (Verbs) by Frequency

