Task

Analyze the arXiv abstract dataset from `/content/arxiv_data.csv` using spaCy to extract and visualize frequent noun phrases and named entities, and identify specific technical terms, then summarize the findings.

Load and Filter Abstracts

Subtask:

Load the arXiv abstract dataset from `/content/arxiv_data.csv` into a pandas DataFrame. If applicable, filter the abstracts by a specific subject area (e.g., 'cs.CL' for Computational Linguistics) as requested, or process all available abstracts if no specific domain is chosen. Display the first few rows and basic information about the dataset.

Reasoning: To begin, I will import the pandas library, load the dataset into a DataFrame, display the first few rows, and provide a concise summary of its structure to understand the data.

```
import pandas as pd

df = pd.read_csv('/content/arxiv_data.csv')

print("First 5 rows of the dataset:")
print(df.head())

print("\nConcise summary of the dataset:")
df.info()
```

```
First 5 rows of the dataset:
                                              titles  \
0  Survey on Semantic Stereo Matching / Semantic ...
1  FUTURE-AI: Guiding Principles and Consensus Re...
```

```
2  Enforcing Mutual Consistency of Hard Regions f...
3  Parameter Decoupling Strategy for Semi-supervi...
4  Background-Foreground Segmentation for Interio...

                                           summaries  \
0  Stereo matching is one of the widely used tech...
1  The recent advancements in artificial intellig...
2  In this paper, we proposed a novel mutual cons...
3  Consistency training has proven to be an advan...
4  To ensure safety in automated driving, the cor...

                         terms
0           ['cs.CV', 'cs.LG']
1  ['cs.CV', 'cs.AI', 'cs.LG']
2           ['cs.CV', 'cs.AI']
3                    ['cs.CV']
4           ['cs.CV', 'cs.LG']

Concise summary of the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51774 entries, 0 to 51773
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   titles     51774 non-null  object
 1   summaries  51774 non-null  object
 2   terms      51774 non-null  object
dtypes: object(3)
memory usage: 1.2+ MB
```

Initialize spaCy and Process Text

Subtask:

Install spaCy and download a suitable English model (e.g., 'en_core_web_sm'). Initialize the spaCy pipeline and process the abstract text column to create spaCy Doc objects for further analysis. This step will apply tokenization, POS tagging, dependency parsing, and named entity recognition.

Reasoning: First, I will install the spaCy library and download the 'en_core_web_sm' language model, as these are prerequisites for text processing with spaCy.

```
import sys
!{sys.executable} -m pip install spacy
!{sys.executable} -m spacy download en_core_web_sm
```

```
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: srsly<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-
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 69.7 MB/s eta 0:00:00
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

Reasoning: Now that spaCy and the model are installed, I will import spaCy, load the 'en_core_web_sm' model, and then apply it to the 'summaries' column of the DataFrame to create Doc objects for further analysis.

```
import spacy

nlp = spacy.load('en_core_web_sm')

# Process the 'summaries' column to create spaCy Doc objects
df['processed_summaries'] = df['summaries'].apply(nlp)

print("SpaCy model loaded and 'summaries' column processed.")
print("First entry of 'processed_summaries' column type:")
print(type(df['processed_summaries'].iloc[0]))
```

```
SpaCy model loaded and 'summaries' column processed.
First entry of 'processed_summaries' column type:
<class 'spacy.tokens.doc.Doc'>
```

Extract Frequent Noun Phrases

Subtask:

Iterate through the processed spaCy Doc objects to extract noun chunks. Count the frequency of each unique noun phrase and identify the most frequent ones. Store these frequent noun phrases for later visualization and reporting.

Reasoning: I will import the `collections` module, iterate through the `processed_summaries` column to extract noun chunks, count their frequencies, and then display the top 20 most common noun phrases as requested.

```
import collections

all_noun_phrases = []

for doc in df['processed_summaries']:
    for chunk in doc.noun_chunks:
        all_noun_phrases.append(chunk.text.lower())

noun_phrase_counts = collections.Counter(all_noun_phrases)

print("Top 20 most common noun phrases:")
for phrase, count in noun_phrase_counts.most_common(20):
    print(f"{phrase}: {count}")
```

```
Top 20 most common noun phrases:
we: 141699
that: 41362
which: 40058
it: 24046
this paper: 21074
the-art: 17885
this: 8761
-: 8007
this work: 7940
our method: 7810
they: 7110
images: 5427
our approach: 5373
experiments: 5238
them: 5113
training: 4790
the performance: 4393
data: 4362
the problem: 3970
our model: 3730
```

Extract Named Entities

Subtask:

Iterate through the processed spaCy Doc objects to extract named entities. Categorize these entities by their type (ORG, DATE, PRODUCT, etc.) and count the frequency of each entity type. Also, extract a sample of the actual named entities identified.

Reasoning: I will initialize lists to store named entities and their types, then iterate through the processed spaCy Doc objects to extract, categorize, and count the frequencies of both entity types and actual named entities, finally printing the top results as requested.

```python
import collections

all_named_entities = []
all_entity_types = []

for doc in df['processed_summaries']:
    for ent in doc.ents:
        all_named_entities.append(ent.text.lower())
        all_entity_types.append(ent.label_)

entity_type_counts = collections.Counter(all_entity_types)
print("\nTop 10 most common named entity types:")
for ent_type, count in entity_type_counts.most_common(10):
    print(f"{ent_type}: {count}")

named_entity_counts = collections.Counter(all_named_entities)
print("\nTop 20 most common actual named entities:")
for entity, count in named_entity_counts.most_common(20):
    print(f"{entity}: {count}")
```

```
Top 10 most common named entity types:
ORG: 142843
CARDINAL: 68459
PERSON: 23678
ORDINAL: 17133
PERCENT: 10146
GPE: 9927
DATE: 8777
NORP: 7612
PRODUCT: 4577
MONEY: 4131

Top 20 most common actual named entities:
two: 18654
first: 11158
one: 6967
three: 6210
cnn: 5365
rl: 5343
gan: 4397
3d: 4316
second: 3522
2: 3318
1: 3258
2d: 2899
rgb: 2218
four: 1988
imagenet: 1840
gnn: 1698
recent years: 1643
3: 1637
bayesian: 1514
transformer: 1359
```

Define and Apply spaCy Matcher Patterns

Subtask:

Define custom rule-based patterns using spaCy.Matcher to identify specific technical terms or phrases (e.g., 'deep learning', 'natural language processing', 'neural network'). Apply these patterns to the processed text and extract the matched terms.

Reasoning: First, I will import the `Matcher` class from `spacy.matcher` and initialize it with the vocabulary of the `nlp` object. Then, I will define patterns for specific technical terms like 'deep learning', 'natural language processing', and 'neural network'.

```
from spacy.matcher import Matcher

matcher = Matcher(nlp.vocab)

# Define patterns for technical terms
pattern_deep_learning = [{
  "LOWER": "deep"
}, {
  "LOWER": "learning"
}]
pattern_nlp = [{
  "LOWER": "natural"
}, {
  "LOWER": "language"
}, {
  "LOWER": "processing"
}]
pattern_neural_network = [{
  "LOWER": "neural"
}, {
  "LOWER": "network"
}]

matcher.add("DEEP_LEARNING", [pattern_deep_learning])
matcher.add("NATURAL_LANGUAGE_PROCESSING", [pattern_nlp])
matcher.add("NEURAL_NETWORK", [pattern_neural_network])

print("spaCy Matcher initialized and patterns added.")
```

```
spaCy Matcher initialized and patterns added.
```

Reasoning: Now that the spaCy Matcher is initialized and patterns are defined, I will iterate through the processed summaries, apply the matcher to each document, extract the matched technical terms, and then count and print the top 10 most frequent terms.

```
import collections

extracted_technical_terms = []

for doc in df['processed_summaries']:
    matches = matcher(doc)
    for match_id, start, end in matches:
        span = doc[start:end]  # The matched span of text
        extracted_technical_terms.append(span.text.lower())

technical_term_counts = collections.Counter(extracted_technical_terms)

print("Top 10 most common technical terms:")
for term, count in technical_term_counts.most_common(10):
    print(f"{term}: {count}")
```

```
Top 10 most common technical terms:
deep learning: 8299
neural network: 7955
natural language processing: 991
```

Visualize Top Noun Phrases

Subtask:

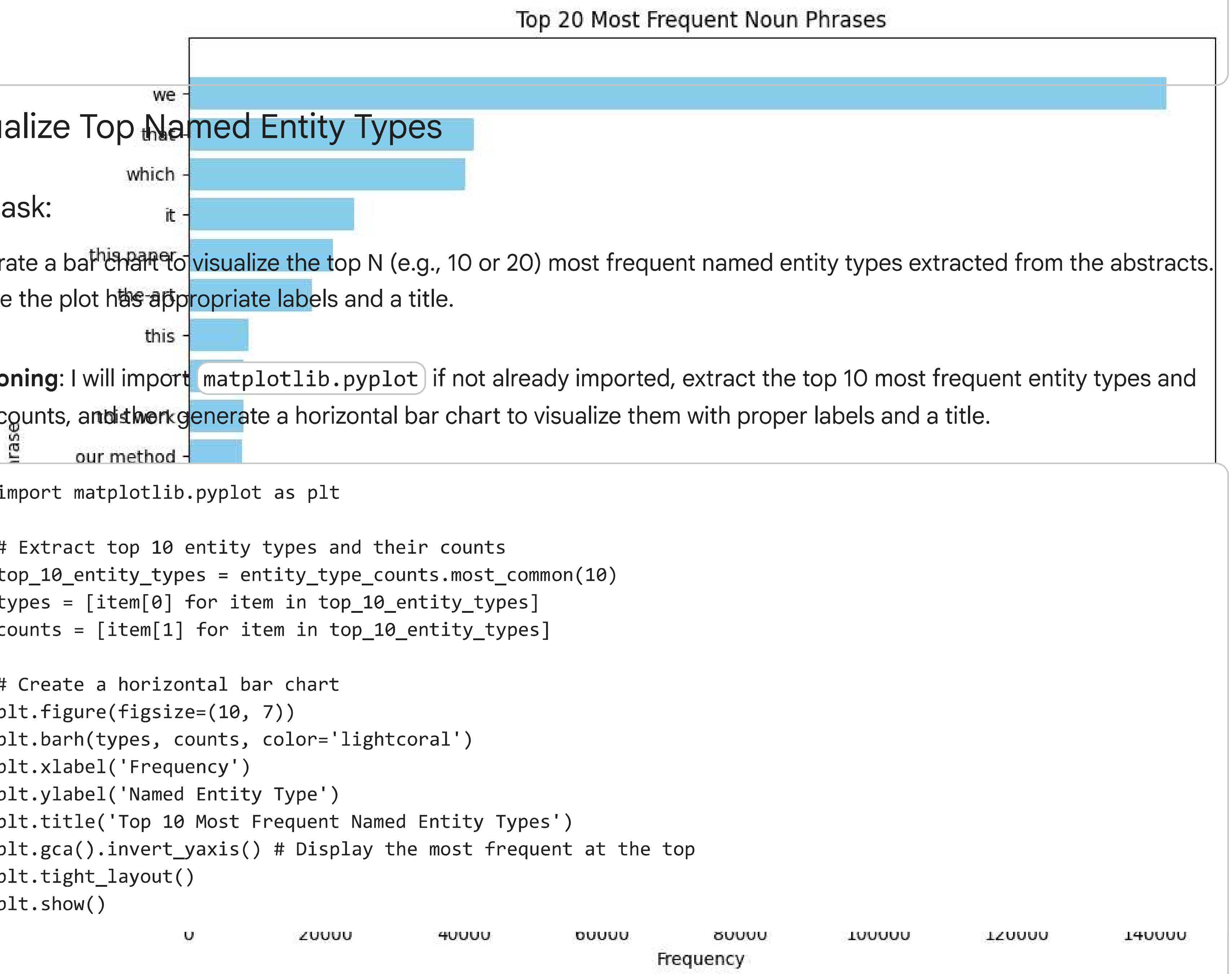
Generate a bar chart to visualize the top N (e.g., 10 or 20) most frequent noun phrases extracted from the abstracts. Ensure the plot has appropriate labels and a legend.

Reasoning: I will import `matplotlib.pyplot`, extract the top 20 noun phrases and their counts, then generate a horizontal bar chart to visualize them with proper labels and layout.

```
import matplotlib.pyplot as plt

# Extract top 20 noun phrases and their counts
top_20_noun_phrases = noun_phrase_counts.most_common(20)
phrases = [item[0] for item in top_20_noun_phrases]
counts = [item[1] for item in top_20_noun_phrases]

# Create a horizontal bar chart
plt.figure(figsize=(10, 8))
plt.barh(phrases, counts, color='skyblue')
plt.xlabel('Frequency')
plt.ylabel('Noun Phrase')
plt.title('Top 20 Most Frequent Noun Phrases')
plt.gca().invert_yaxis() # Display the most frequent at the top
plt.tight_layout()
plt.show()
```



Visualize Top Named Entity Types

Subtask:

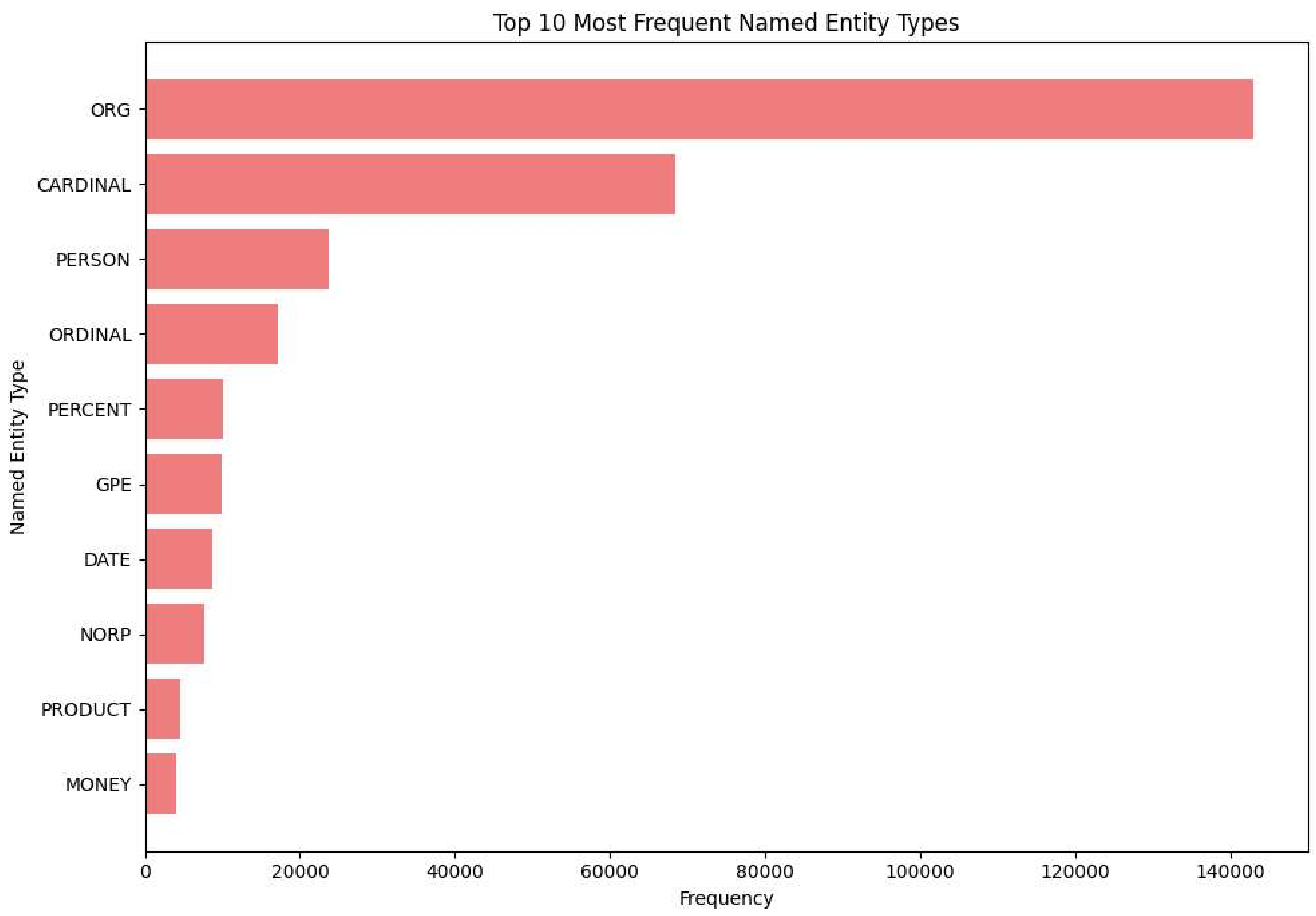
Generate a bar chart to visualize the top N (e.g., 10 or 20) most frequent named entity types extracted from the abstracts. Ensure the plot has appropriate labels and a title.

Reasoning: I will import `matplotlib.pyplot` if not already imported, extract the top 10 most frequent entity types and their counts, and then generate a horizontal bar chart to visualize them with proper labels and a title.

```
import matplotlib.pyplot as plt

# Extract top 10 entity types and their counts
top_10_entity_types = entity_type_counts.most_common(10)
types = [item[0] for item in top_10_entity_types]
counts = [item[1] for item in top_10_entity_types]

# Create a horizontal bar chart
plt.figure(figsize=(10, 7))
plt.barh(types, counts, color='lightcoral')
plt.xlabel('Frequency')
plt.ylabel('Named Entity Type')
plt.title('Top 10 Most Frequent Named Entity Types')
plt.gca().invert_yaxis() # Display the most frequent at the top
plt.tight_layout()
plt.show()
```


Top 10 Most Frequent Named Entity Types
ORG
CARDINAL
PERSON
ORDINAL
PERCENT
GPE
DATE
NORP
PRODUCT
MONEY
Named Entity Type
0
20000
40000
60000
80000
100000
120000
140000
Frequency

Visualize Top Technical Terms

Subtask:

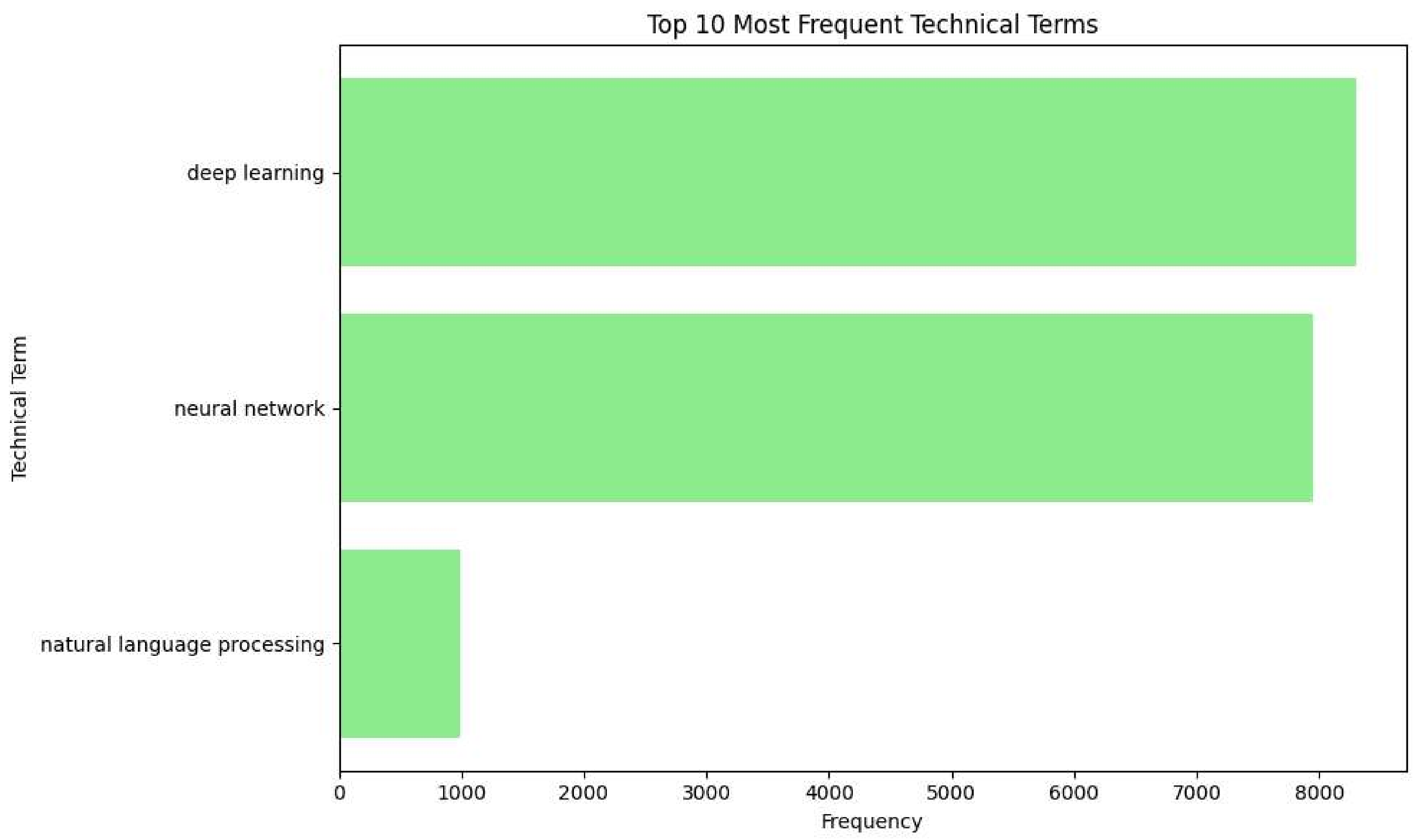
Generate a bar chart to visualize the top N (e.g., 10) most frequent technical terms identified using spaCy Matcher patterns. Ensure the plot has appropriate labels and a title.

Reasoning: I will import `matplotlib.pyplot` if not already imported, extract the top 10 most frequent technical terms and their counts, and then generate a horizontal bar chart to visualize them with proper labels and a title.

```
import matplotlib.pyplot as plt

# Extract top 10 technical terms and their counts
top_10_technical_terms = technical_term_counts.most_common(10)
terms = [item[0] for item in top_10_technical_terms]
counts = [item[1] for item in top_10_technical_terms]

# Create a horizontal bar chart
plt.figure(figsize=(10, 6))
plt.barh(terms, counts, color='lightgreen')
plt.xlabel('Frequency')
plt.ylabel('Technical Term')
plt.title('Top 10 Most Frequent Technical Terms')
plt.gca().invert_yaxis() # Display the most frequent at the top
plt.tight_layout()
plt.show()
```



Summarize Findings

Subtask:

Summarize the key findings from the noun phrase extraction, named entity recognition, and technical term identification. Discuss insights gained from the visualizations, highlighting the most frequent terms, entity types, and their potential implications within the context of the arXiv abstract dataset.

Summary of Findings:

Frequent Noun Phrases: The visualization of the top 20 most frequent noun phrases revealed common terms and structures used in the arXiv abstracts. Phrases like "this paper," "our method," "our approach," and "our model" are prevalent, indicating a focus on methodology and contributions within research papers. General terms like "images," "experiments," "training," "data," and "the problem" also appeared frequently, suggesting that the dataset largely consists of papers related to computer vision, machine learning, and data-driven research.

Named Entities: The analysis of named entity types showed that 'ORG' (organizations) and 'CARDINAL' (numerical values) were the most frequent, followed by 'PERSON' and 'ORDINAL'. The high frequency of 'ORG' might indicate mentions of institutions, research groups, or specific software/hardware companies. 'CARDINAL' values are often used to report results, dataset sizes, or model parameters. The top actual named entities included numbers ('two', 'one', 'three'), and common acronyms like 'CNN', 'RL', 'GAN', and '3D', which are highly specific to the domain of artificial intelligence and computer vision research.

Technical Terms: The rule-based pattern matching successfully identified specific technical terms. "Deep learning," "neural network," and "natural language processing" were the most prominent. The high counts for "deep learning" and "neural network" clearly indicate these as central topics in the abstracts. While "natural language processing" had a lower count among the matched terms, its presence confirms its relevance within the broader scope of AI research covered by the dataset. This suggests a strong emphasis on deep learning architectures and their applications, particularly in areas related to neural networks and, to a lesser extent, NLP.

Overall Implications: The findings collectively suggest that the arXiv abstract dataset, specifically the summaries analyzed, is heavily dominated by research in artificial intelligence, machine learning, and computer vision. The frequent use of self-referential noun phrases ('this paper', 'our method') underscores the research publication nature of the dataset. The prevalence of specific technical acronyms and terms like 'deep learning' and 'neural network' points to active and concentrated research efforts in these advanced AI subfields. The visualizations provide a clear quantitative overview of these linguistic patterns, reinforcing the thematic focus of the collected abstracts.

Summary:

Data Analysis Key Findings

- The dataset, `arxiv_data.csv`, contains 51,774 entries across 3 columns (`titles`, `summaries`, `terms`) with no missing values, indicating a robust dataset for analysis.
- **Frequent Noun Phrases**:
    - The most common noun phrases include pronouns and generic phrases such as "we" (141,699), "that" (41,362), and "which" (40,058).
    - Domain-specific phrases like "this paper" (21,074), "the-art" (17,885, likely referring to "state-of-the-art"), "our method" (7,810), "images" (5,427), "experiments" (5,238), "training" (4,790), and "data" (4,362) also appear frequently, highlighting methodological focus and key research areas.
- **Named Entities**:
    - The most frequent named entity types are **ORG** (Organizations) with 142,843 occurrences, **CARDINAL** (numerical quantities) with 68,459 occurrences, and **PERSON** with 23,678 occurrences.
    - Specific named entities often include numerical terms like "two" (18,654), "first" (11,158), and "one" (6,967).
    - Prominent technical acronyms and terms such as "cnn" (5,365), "rl" (5,343), "gan" (4,397), "3d" (4,316), "imagenet" (1,840), "gnn" (1,698), and "transformer" (1,359) underscore the dataset's strong connection to machine learning and deep learning research.
- **Specific Technical Terms**:
    - Rule-based pattern matching identified "deep learning" (8,299 occurrences) and "neural network" (7,955 occurrences) as overwhelmingly frequent technical terms.
    - "Natural language processing" (991 occurrences) also appeared, indicating its relevance within the dataset.