```
# Install necessary libraries for data processing, NLP, and visualization
!pip install spacy pandas matplotlib seaborn wordcloud
# Download the English small language model for spaCy
!python -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: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-packages (1.9.5)
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: srsly<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: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (4.67.1)
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,>=1.7.4 in /usr/local/lib/python3.12/dist-packages (from spacy
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: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.3.1)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<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->spac
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0
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->sp
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
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packages (from typer-slim<1.0.0,>=0.3.0->spa
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=
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.
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
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-3.8.0-py3-no
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 104.2 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.
```

```
import pandas as pd # Used for data manipulation and analysis
import spacy # Natural Language Processing library
from spacy.matcher import Matcher # For rule-based token matching in spaCy
from collections import Counter # For counting hashable objects
import matplotlib.pyplot as plt # For creating static, interactive, and animated visualizations
from wordcloud import WordCloud # For generating word cloud images
```

```
# Load dataset from 'arxiv_data.csv'
# Using engine='python' and on_bad_lines='skip' to handle potential parsing errors in the CSV file
df = pd.read_csv("arxiv_data.csv", engine='python', on_bad_lines='skip')

# Display the first 5 rows of the DataFrame to inspect the data
df.head()
```

	titles	summaries	terms
0	Survey on Semantic Stereo Matching / Semantic ...	Stereo matching is one of the widely used tech...	['cs.CV', 'cs.LG']
1	FUTURE-AI: Guiding Principles and Consensus Re...	The recent advancements in artificial intellig...	['cs.CV', 'cs.AI', 'cs.LG']
2	Enforcing Mutual Consistency of Hard Regions f...	In this paper, we proposed a novel mutual cons...	['cs.CV', 'cs.AI']
3	Parameter Decoupling Strategy for Semi-supervi...	Consistency training has proven to be an advan...	['cs.CV']
4	Background-Foreground Segmentation for Interio...	To ensure safety in automated driving, the cor...	['cs.CV', 'cs.LG']

Next steps: Generate code with df New interactive sheet

```
# Filter the DataFrame to include only papers related to 'cs' (Computer Science) based on the 'terms' column
df_cs = df[df['terms'].str.contains('cs', na=False)]

# Select the 'summaries' (abstracts) from the filtered DataFrame and take the first 100 entries
# Convert these summaries into a list for further NLP processing
abstracts = df_cs['summaries'].head(100).tolist()

# Print the total number of abstracts selected for processing
print("Number of abstracts:", len(abstracts))
```

```
Number of abstracts: 100
```

```
# Load the pre-trained English small language model from spaCy
nlp = spacy.load("en_core_web_sm")

# Process the first abstract from the 'abstracts' list using the loaded spaCy model
doc = nlp(abstracts[0])
# Print the first 50 tokens of the processed document to inspect the output
print(doc[:50])
```

```
Stereo matching is one of the widely used techniques for inferring depth from
stereo images owing to its robustness and speed. It has become one of the major
topics of research since it finds its applications in autonomous driving,
robotic navigation, 3D reconstruction,
```

```
# Extract tokens from the processed document, filtering out stopwords and punctuation
tokens = [token.text for token in doc if not token.is_stop and not token.is_punct]
# Display the first 20 extracted tokens
tokens[:20]
```

```
['Stereo',
 'matching',
 'widely',
 'techniques',
 'inferring',
 'depth',
 '\n',
 'stereo',
 'images',
 'owing',
 'robustness',
 'speed',
 'major',
 '\n',
 'topics',
 'research',
 'finds',
 'applications',
 'autonomous',
 'driving']
```

```
noun_phrases = [] # Initialize an empty list to store noun phrases

# Iterate through each abstract in the 'abstracts' list
for text in abstracts:
    doc = nlp(text) # Process the abstract text with the spaCy NLP model
    # Extract noun chunks (noun phrases) from the processed document
    for chunk in doc.noun_chunks:
        noun_phrases.append(chunk.text.lower()) # Convert to lowercase and add to the list

# Count the frequency of each noun phrase
np_freq = Counter(noun_phrases)
# Get the 10 most common noun phrases and their counts
top_nps = np_freq.most_common(10)

# Display the top 10 most frequent noun phrases
top_nps
```

```
[('we', 265),
 ('which', 74),
 ('that', 73),
 ('it', 72),
 ('the-art', 42),
 ('this paper', 34),
 ('medical image segmentation', 25),
 ('our method', 25),
 ('this work', 24),
 ('image segmentation', 22)]
```

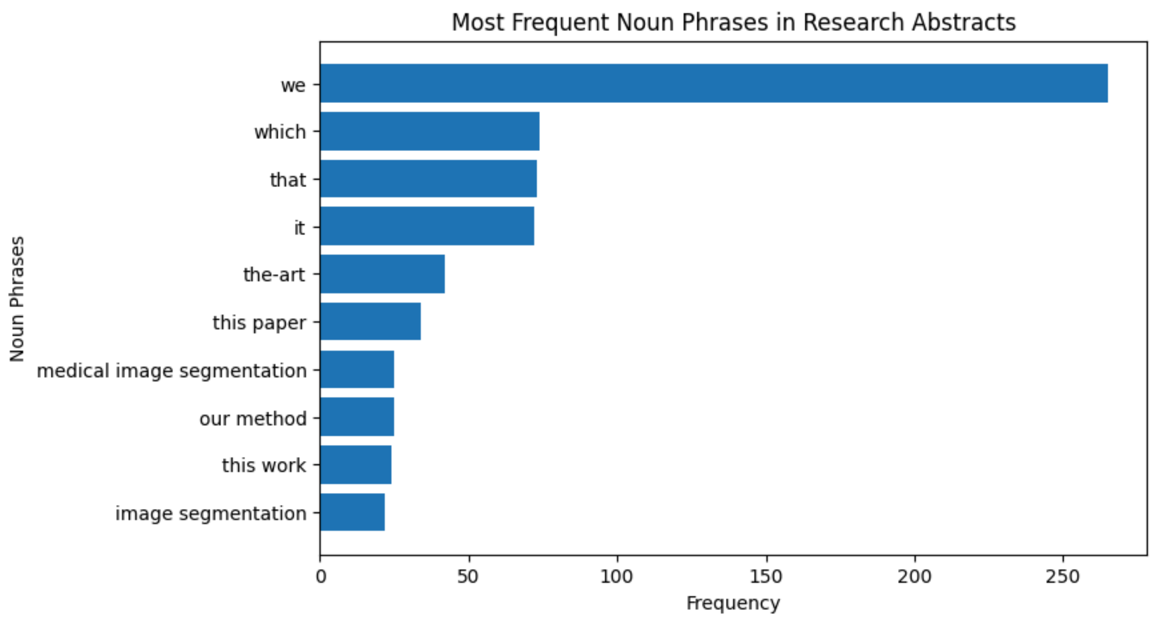
```
entities = [] # Initialize an empty list to store named entity labels

# Iterate through each abstract in the 'abstracts' list
for text in abstracts:
    doc = nlp(text) # Process the abstract text with the spaCy NLP model
    # Extract named entities from the processed document
    for ent in doc.ents:
        entities.append(ent.label_) # Append the label of each entity to the list

# Count the frequency of each named entity label
entity_freq = Counter(entities)
# Display the frequency of each named entity type
entity_freq
```

```
Counter({'DATE': 13,
         'GPE': 21,
         'CARDINAL': 132,
         'NORP': 15,
         'ORG': 247,
         'ORDINAL': 37,
         'WORK_OF_ART': 2,
         'PERSON': 31,
         'PERCENT': 19,
         'PRODUCT': 6,
         'MONEY': 4,
         'TIME': 2,
         'LOC': 1,
         'LAW': 1,
         'EVENT': 1,
         'FAC': 3})
```

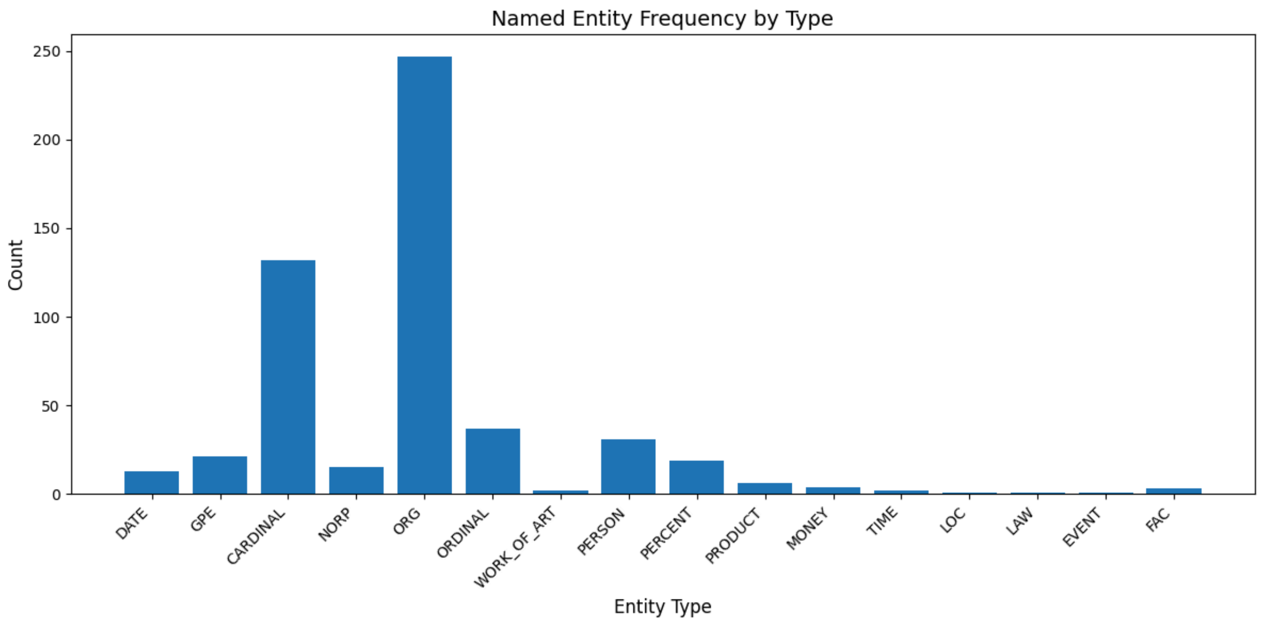
```
matcher = Matcher(nlp.vocab) # Initialize a Matcher object with the spaCy vocabulary

# Define patterns for common technical terms: 'deep learning', 'neural network', 'machine learning'
pattern1 = [{"LOWER": "deep"}, {"LOWER": "learning"}]
pattern2 = [{"LOWER": "neural"}, {"LOWER": "network"}]
pattern3 = [{"LOWER": "machine"}, {"LOWER": "learning"}]

# Add these patterns to the matcher under the label "TECH_TERMS"
matcher.add("TECH_TERMS", [pattern1, pattern2, pattern3])

matches = [] # Initialize an empty list to store found matches

# Iterate through each abstract in the 'abstracts' list
for text in abstracts:
    doc = nlp(text) # Process the abstract text with the spaCy NLP model
    # Apply the matcher to the document to find all occurrences of the defined patterns
    for match_id, start, end in matcher(doc):
        matches.append(doc[start:end].text) # Append the matched text to the list

# Count the frequency of each matched technical term
Counter(matches)
```

```
Counter({'neural network': 9,
         'deep learning': 15,
         'Deep learning': 6,
         'machine learning': 7,
         'Machine learning': 3,
         'Neural Network': 1,
         'Deep Learning': 2})
```

```
# Extract noun phrases (labels) and their counts from the 'top_nps' list for plotting
phrases = [item[0] for item in top_nps]
counts = [item[1] for item in top_nps]

# Create a horizontal bar chart to visualize the frequency of the most common noun phrases
plt.figure(figsize=(8, 5)) # Set the figure size for better readability
plt.barh(phrases, counts) # Create the horizontal bar chart
plt.xlabel("Frequency") # Label for the x-axis
plt.ylabel("Noun Phrases") # Label for the y-axis
plt.title("Most Frequent Noun Phrases in Research Abstracts") # Title of the plot
plt.gca().invert_yaxis() # Invert the y-axis to display the most frequent phrase at the top
plt.show() # Display the plot
```



```
# Prepare data for plotting: extract entity types (keys) and their counts (values) from the 'entity_freq' Counter
entity_types = list(entity_freq.keys())
entity_counts = list(entity_freq.values())

# Create a bar chart to visualize the frequency of different named entity types
plt.figure(figsize=(12, 6)) # Set the figure size for better readability

plt.bar(entity_types, entity_counts) # Create the bar chart

# Improve label visibility and overall plot aesthetics
plt.xlabel("Entity Type", fontsize=12) # Label for the x-axis with increased font size
plt.ylabel("Count", fontsize=12) # Label for the y-axis with increased font size
plt.title("Named Entity Frequency by Type", fontsize=14) # Title of the plot with increased font size

plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels by 45 degrees for clarity if they overlap
plt.tight_layout()                   # Adjust plot parameters for a tight layout, preventing labels from being cut off

plt.show() # Display the plot
```


Named Entity Frequency by Type
Count
DATE
GPE
CARDINAL
NORP
ORG
ORDINAL
WORK_OF_ART
PERSON
PERCENT
PRODUCT
MONEY
TIME
LOC
LAW
EVENT
FAC
Entity Type