```
!pip install spacy pandas matplotlib seaborn
!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: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packages (from spacy) (3
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0
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) (2.0.1
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
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.
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/dist-packages (from
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
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,
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pand
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2
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.
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packages (from typer-slim<1.0.0,>=0.3.0
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0
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->wease
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-p
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 63.8 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
import spacy
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns
from spacy.matcher import Matcher
```

```
# Load the arXiv abstracts dataset and preview its structure
df = pd.read_csv("/content/arxiv_data.csv")

print("Dataset shape:", df.shape)
df.head()
```

```
Dataset shape: (51774, 3)
```

| | 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'] |

```
# Extract non-empty research abstracts from the dataset and store them as a list
abstracts = df['summaries'].dropna().tolist()
```

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

```
# Process the first 500 abstracts through the spaCy pipeline for efficient NLP analysis
docs = list(nlp.pipe(abstracts[:500]))  # limit for performance
```

```
# Tokenize the first abstract and extract the first 30 tokens excluding punctuation
doc = docs[0]

tokens = [token.text for token in doc if not token.is_punct]
tokens[:30]
```

```
['Stereo',
 'matching',
 'is',
 'one',
 'of',
 'the',
 'widely',
 'used',
 'techniques',
 'for',
 'inferring',
 'depth',
 'from',
 '\n',
 'stereo',
 'images',
 'owing',
 'to',
 'its',
 'robustness',
 'and',
 'speed',
 'It',
 'has',
 'become',
 'one',
 'of',
 'the',
 'major',
 '\n']
```

```
# Extract noun phrases from all processed abstracts and find the top 10 most frequent ones
noun_phrases = []

for doc in docs:
    for chunk in doc.noun_chunks:
        noun_phrases.append(chunk.text.lower())

np_freq = Counter(noun_phrases)
top_noun_phrases = np_freq.most_common(10)

top_noun_phrases
```

```
[('we', 1295),
 ('which', 425),
 ('that', 326),
 ('it', 281),
 ('this paper', 195),
 ('the-art', 191),
 ('image segmentation', 148),
 ('our method', 115),
 ('-', 99),
 ('this work', 98)]
```

```
# Extract named entities from abstracts and compute frequency of each entity type
entities = []

for doc in docs:
    for ent in doc.ents:
        entities.append((ent.text, ent.label_))

entity_df = pd.DataFrame(entities, columns=["Entity", "Label"])
entity_freq = entity_df['Label'].value_counts()

entity_freq
```

| Label | count |
|---|---|
| ORG | 1353 |
| CARDINAL | 761 |
| PERSON | 184 |
| PERCENT | 150 |
| ORDINAL | 143 |
| GPE | 113 |
| DATE | 80 |
| NORP | 68 |
| PRODUCT | 35 |
| MONEY | 26 |
| WORK_OF_ART | 16 |
| FAC | 14 |
| QUANTITY | 9 |
| LOC | 7 |
| TIME | 7 |
| LAW | 4 |
| EVENT | 2 |

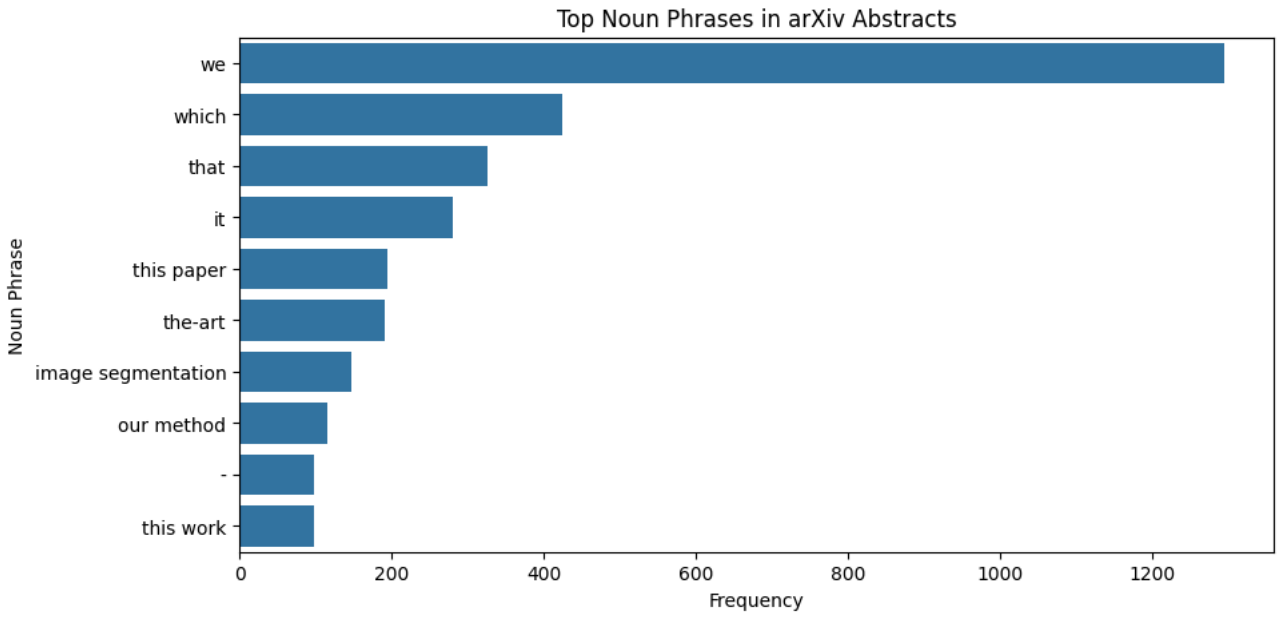
**dtype:** int64

```
# Define rule-based patterns using spaCy Matcher to identify technical terms in abstracts
matcher = Matcher(nlp.vocab)

pattern1 = [
    {"POS": "ADJ"},
    {"POS": "NOUN"}
]

pattern2 = [
    {"POS": "NOUN"},
    {"POS": "NOUN"}
]

matcher.add("TECH_TERM", [pattern1, pattern2])
```

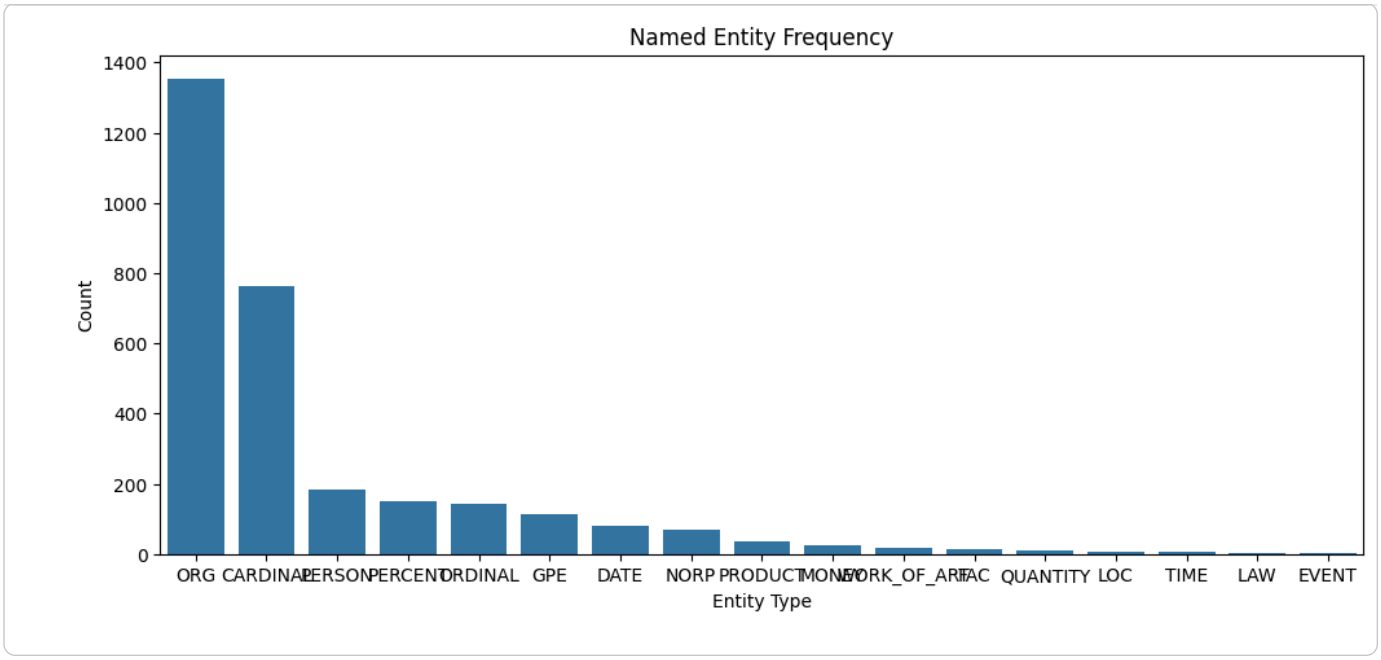
```
# Apply matcher patterns to extract frequent technical terms from the abstracts
matched_terms = []

for doc in docs:
    matches = matcher(doc)
    for match_id, start, end in matches:
        matched_terms.append(doc[start:end].text.lower())

Counter(matched_terms).most_common(10)
```

```
[('image segmentation', 525),
 ('medical image', 202),
 ('semantic segmentation', 122),
 ('deep learning', 117),
 ('neural networks', 77),
 ('segmentation tasks', 69),
 ('neural network', 58),
 ('experimental results', 56),
 ('computer vision', 51),
 ('medical images', 50)]
```

```
# Visualize the top 10 most frequent noun phrases using a bar chart
phrases, counts = zip(*top_noun_phrases)

plt.figure(figsize=(10,5))
sns.barplot(x=list(counts), y=list(phrases))
plt.title("Top Noun Phrases in arXiv Abstracts")
plt.xlabel("Frequency")
plt.ylabel("Noun Phrase")
plt.show()
```




```
# Visualize the frequency distribution of named entity types using a bar chart
plt.figure(figsize=(12,5))
sns.barplot(x=entity_freq.index, y=entity_freq.values)
plt.title("Named Entity Frequency")
plt.xlabel("Entity Type")
plt.ylabel("Count")
plt.show()
```


Named Entity Frequency
Count
1400
1200
1000
800
600
400
200
0
ORG
CARDINAL
PERSON
PERCENT
ORDINAL
GPE
DATE
NORP
PRODUCT
MONEY
WORK_OF_ART
FAC
QUANTITY
LOC
TIME
LAW
EVENT
Entity Type