Installing libraries:

```
!pip install -U spacy
!python -m spacy download en_core_web_sm

!pip install matplotlib seaborn

import spacy
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from spacy.matcher import Matcher
```

```
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: 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: 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 93.0 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.
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: 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: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
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) (3.3.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.po
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib)
```

Abstract dataset:

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os


directory_path = '/content/drive/My Drive/NLP DATASETS/'


if os.path.exists(directory_path):
    print(f"Contents of '{directory_path}':")
    for item in os.listdir(directory_path):
        print(item)
else:
    print(f"Directory not found: '{directory_path}'\nPlease ensure Google Drive is mounted and the path is correct.")
```

```
Contents of '/content/drive/My Drive/NLP DATASETS/':
arxiv_data.csv
```

```
import pandas as pd
import os


if not os.path.exists('arxiv_data.csv'):
    !unzip -o arxiv_data.csv.zip

df = pd.read_csv("arxiv_data.csv")

cs_ai_mask = df["terms"].str.contains("cs.", na=False)
df_cs = df[cs_ai_mask].copy()

df_cs = df_cs[["titles", "summaries"]].dropna()

abstracts = df_cs["summaries"].head(100).tolist()

print(f"Successfully loaded {len(abstracts)} abstracts.")
```

```
Successfully loaded 100 abstracts.
```

Tokenization:

```
nlp = spacy.load('en_core_web_sm')
docs = list(nlp.pipe(abstracts))
```

Extract Frequent Noun Phrases:

```
all_noun_phrases = []
raw_noun_chunks_count = 0

for doc in docs:
    for chunk in doc.noun_chunks:
        raw_noun_chunks_count += 1

        phrase = chunk.text.strip().lower()

        if len(phrase.split()) >= 2 and len(phrase) > 3:
            all_noun_phrases.append(phrase)

print(f"Total raw noun chunks found: {raw_noun_chunks_count}")
print(f"Total noun phrases after filtering: {len(all_noun_phrases)}")


phrase_freq = Counter(all_noun_phrases)
top_phrases = phrase_freq.most_common(20)


print("Top Noun Phrases:")
for phrase, count in top_phrases:
    print(f"  {phrase:<25} → {count}")
```

```
Total raw noun chunks found: 4854
Total noun phrases after filtering: 3551
Top Noun Phrases:
  this paper                → 34
  medical image segmentation → 25
  our method                → 25
  this work                 → 24
  image segmentation        → 22
  semantic segmentation     → 20
  the performance           → 15
  the model                 → 12
  our approach              → 10
```

```
  deep learning             → 9
  unlabeled data            → 8
  medical images            → 8
  the effectiveness         → 7
  the network               → 7
  extensive experiments     → 7
  an image                  → 7
  the segmentation          → 7
  a set                     → 7
  medical imaging           → 6
  the problem               → 6
```

Named entities (ORG, DATE, PRODUCT, etc.):

```
from collections import Counter

interested_labels = ["ORG", "DATE", "PRODUCT", "GPE", "LOC", "PERSON", "EVENT", "NORP"]

all_named_entities = []

for doc in docs:
    for ent in doc.ents:
        if ent.label_ in interested_labels:
            entity_text = ent.text.strip().lower()
            all_named_entities.append((entity_text, ent.label_))
entity_freq = Counter(all_named_entities)
top_entities = entity_freq.most_common(20)

print("Top Named Entities:")
print(f"  {'Entity':<30} {'Type':<10} {'Count'}")
print(f"  {'-'*30} {'-'*10} {'-'*5}")
for (entity, label), count in top_entities:
    print(f"  {entity:<30} {label:<10} {count}")
```

```
Top Named Entities:
  Entity                         Type       Count
  ------------------------------ ---------- -----
  3d                             ORG        15
  cnn                            ORG        9
  u-net                          ORG        8
  transformer                    ORG        7
  ssl                            ORG        7
  ct                             ORG        7
  recent years                   DATE       6
  ai                             GPE        5
  nas                            ORG        5
  fss                            ORG        5
  sas                            ORG        5
  linear                         ORG        4
  uda                            ORG        4
  mia                            ORG        4
  la                             GPE        4
  eht                            ORG        3
  mps                            ORG        3
  pmtrans                        ORG        3
  cityscapes                     GPE        3
  bayesian                       NORP       3
```

Use spaCy Matcher to identify technical term patterns:

```
from spacy.matcher import Matcher

matcher = Matcher(nlp.vocab)

pattern1 = [{"POS": "ADJ", "OP": "*"}, {"POS": "NOUN", "OP": "+"}]
pattern2 = [{"POS": "NOUN", "OP": "+"}, {"POS": "NOUN", "OP": "+"}]
pattern3 = [{"POS": "PROPN", "OP": "+"}, {"POS": "NOUN", "OP": "+"}]

matcher.add("TECHNICAL_TERM", [pattern1, pattern2, pattern3])

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

term_freq = Counter(technical_terms)
top_technical_terms = term_freq.most_common(20)

print("Top Technical Term Patterns:")
```

```
for term, count in top_technical_terms:
    print(f"  {term:<30} → {count}")
```

```
Top Technical Term Patterns:
  segmentation                   → 305
  image                          → 233
  data                           → 135
  image segmentation             → 114
  learning                       → 83
  images                         → 79
  performance                    → 77
  network                        → 74
  training                       → 74
  methods                        → 73
  model                          → 69
  method                         → 69
  medical image                  → 62
  datasets                       → 54
  state                          → 52
  information                    → 51
  framework                      → 49
  art                            → 48
  results                        → 47
  medical image segmentation     → 46
```

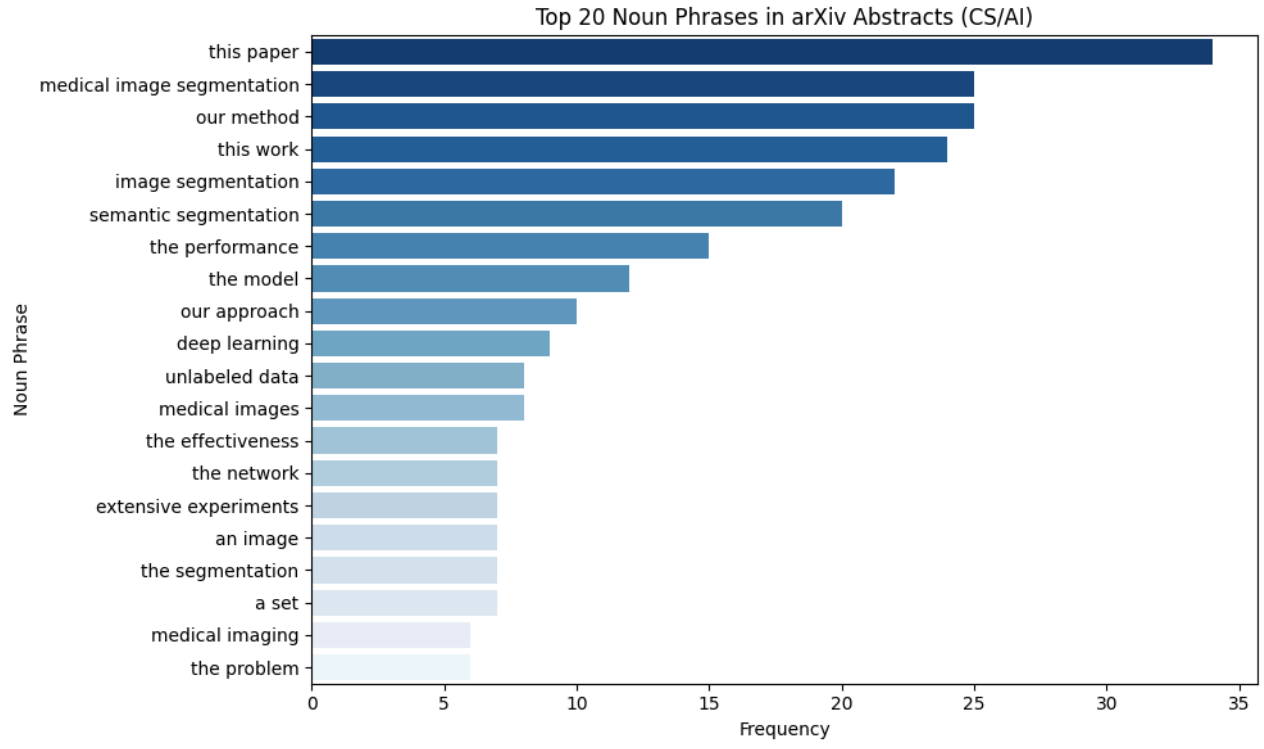
```
top_phrases_df = pd.DataFrame(top_phrases, columns=["Phrase", "Count"])

plt.figure(figsize=(10, 6))
sns.barplot(data=top_phrases_df, y="Phrase", x="Count", palette="Blues_r")
plt.title("Top 20 Noun Phrases in arXiv Abstracts (CS/AI)")
plt.xlabel("Frequency")
plt.ylabel("Noun Phrase")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-697529154.py:6: FutureWarning:

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

  sns.barplot(data=top_phrases_df, y="Phrase", x="Count", palette="Blues_r")
```




```
from collections import Counter

label_counts = Counter(label for entity, label in all_named_entities)

label_counts_df = pd.DataFrame(label_counts.most_common(), columns=["Label", "Count"])

plt.figure(figsize=(10, 6))
sns.barplot(data=label_counts_df, y="Label", x="Count", palette="Greens_r")
plt.title("Named Entity Counts by Label")
```

```
plt.xlabel("Frequency")
plt.ylabel("NER Label")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-4229480176.py:10: FutureWarning:

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

  sns.barplot(data=label_counts_df, y="Label", x="Count", palette="Greens_r")
```