```
!pip install spacy pandas matplotlib collections
!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)
ERROR: Could not find a version that satisfies the requirement collections (from versions: none)
ERROR: No matching distribution found for collections
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_
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 82.9 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 spacy
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from spacy.matcher import Matcher
```

```
df = pd.read_csv("/content/arxiv_data.csv")
# df = df[df['category'].str.contains('cs', na=False)]  # select CS domain
# print(df.columns)
texts = df['summaries'].dropna().head(200)  # subset for lab
```

```
nlp = spacy.load("en_core_web_sm")
```

```
doc = nlp(texts.iloc[0])

tokens = [token.text for token in doc]
print(tokens[:30])
```

```
['Stereo', 'matching', 'is', 'one', 'of', 'the', 'widely', 'used', 'techniques', 'for', 'inferring
```

```
noun_phrases = []

for text in texts:
    doc = nlp(text)
    noun_phrases.extend([chunk.text.lower() for chunk in doc.noun_chunks])

np_freq = Counter(noun_phrases)
top_nps = np_freq.most_common(10)
top_nps
```

```
[('we', 540),
 ('which', 172),
 ('that', 144),
```

```
 ('it', 120),
 ('this paper', 74),
 ('the-art', 72),
 ('our method', 50),
 ('image segmentation', 47),
 ('this work', 47),
 ('medical image segmentation', 37)]
```

```
entities = []

for text in texts:
    doc = nlp(text)
    entities.extend([(ent.text, ent.label_) for ent in doc.ents])

entity_freq = Counter([label for _, label in entities])
entity_freq
```

```
Counter({'DATE': 28,
         'GPE': 44,
         'CARDINAL': 284,
         'NORP': 42,
         'ORG': 525,
         'ORDINAL': 67,
         'WORK_OF_ART': 6,
         'PERSON': 60,
         'PERCENT': 54,
         'PRODUCT': 15,
         'MONEY': 12,
         'TIME': 3,
         'LOC': 4,
         'LAW': 2,
         'EVENT': 1,
         'FAC': 4,
         'QUANTITY': 2})
```

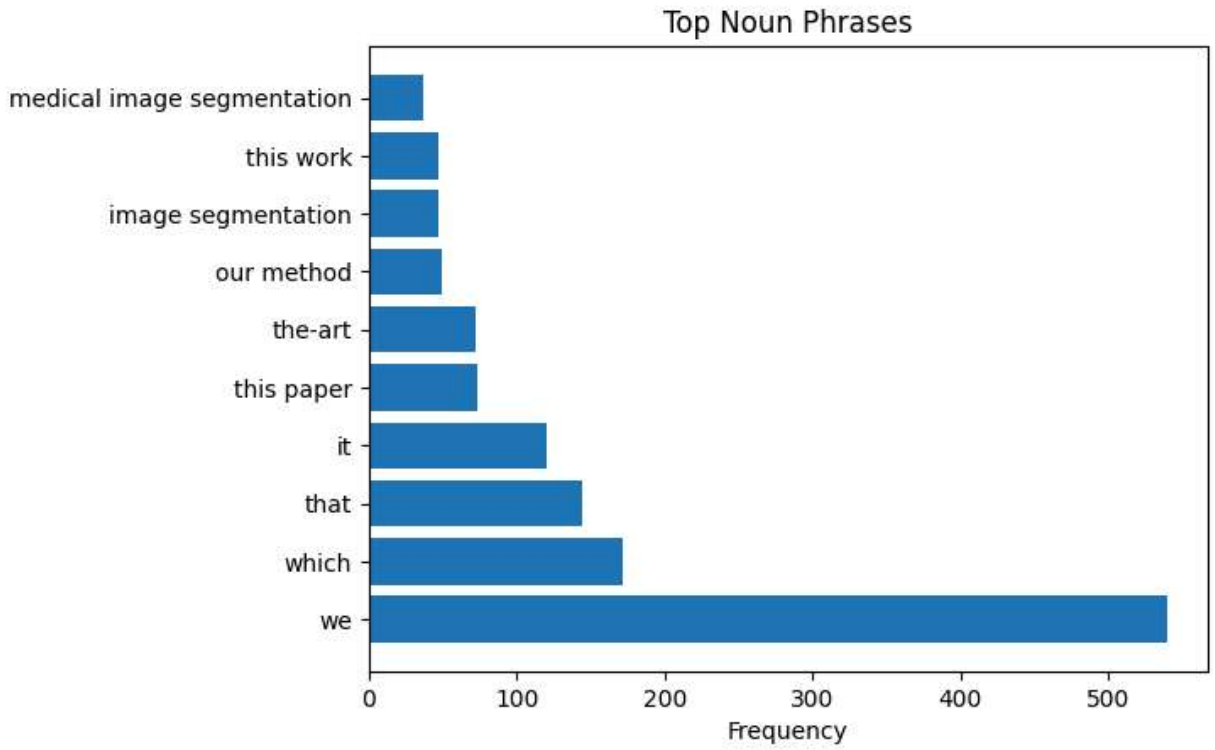
```
matcher = Matcher(nlp.vocab)

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

matcher.add("TECH_TERM", [pattern])
```

```
matches_found = []

for text in texts:
    doc = nlp(text)
    matches = matcher(doc)
    for match_id, start, end in matches:
        matches_found.append(doc[start:end].text.lower())

Counter(matches_found).most_common(10)
```

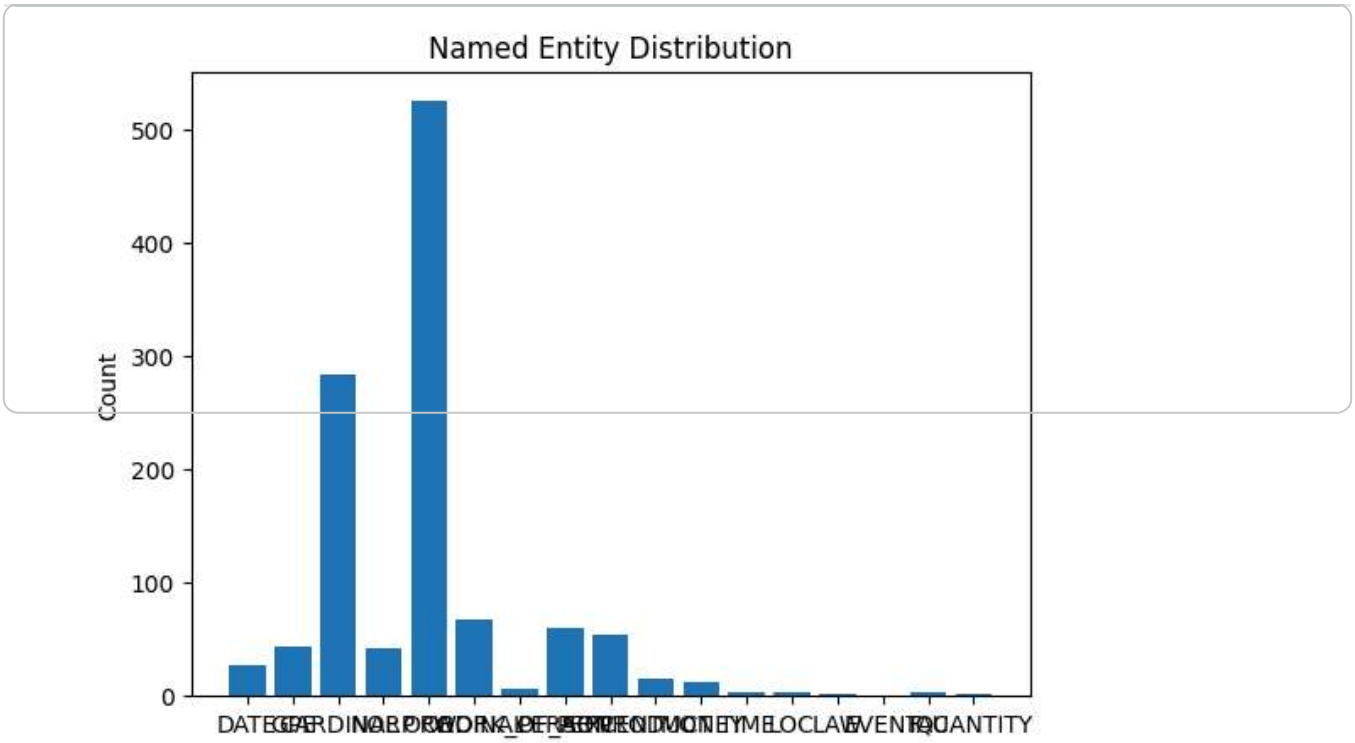
```
[('medical image segmentation', 66),
 ('semantic image segmentation', 12),
 ('medical image analysis', 7),
 ('deep reinforcement learning', 5),
```

```
 ('accurate image segmentation', 4),
 ('interactive segmentation methods', 3),
 ('urban scene images', 3),
 ('deep learning methods', 3),
 ('deep neural networks', 3),
 ('automatic image segmentation', 3)]
```

```
labels, values = zip(*top_nps)

plt.figure()
plt.barh(labels, values)
plt.title("Top Noun Phrases")
plt.xlabel("Frequency")
plt.show()
```




```
labels, values = zip(*entity_freq.items())

plt.figure()
plt.bar(labels, values)
plt.title("Named Entity Distribution")
plt.ylabel("Count")
plt.show()
```


Named Entity Distribution
Count
500
400
300
200
100
0
DATE
QUANTITY