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

```
df = pd.read_csv("/content/arxiv_data 5.csv")
```

```
cs_df = df[df['titles'].str.contains("cs", na=False)]
cs_df[['titles']].head()
```

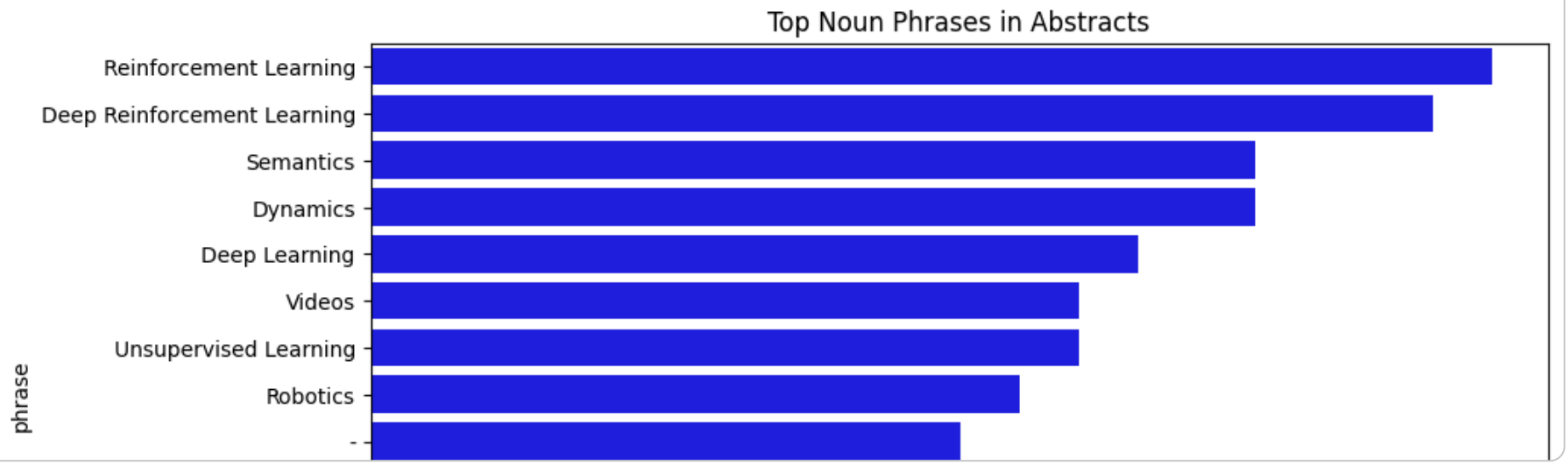
	titles
93	Inter Extreme Points Geodesics for Weakly Supe...
103	Semantics-aware Multi-modal Domain Translation...
124	RLCorrector: Reinforced Proofreading for Conne...
226	PointFlow: Flowing Semantics Through Points fo...
317	Color Image Segmentation Metrics

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

```
docs = list(nlp.pipe(cs_df['titles'].astype(str)))
```

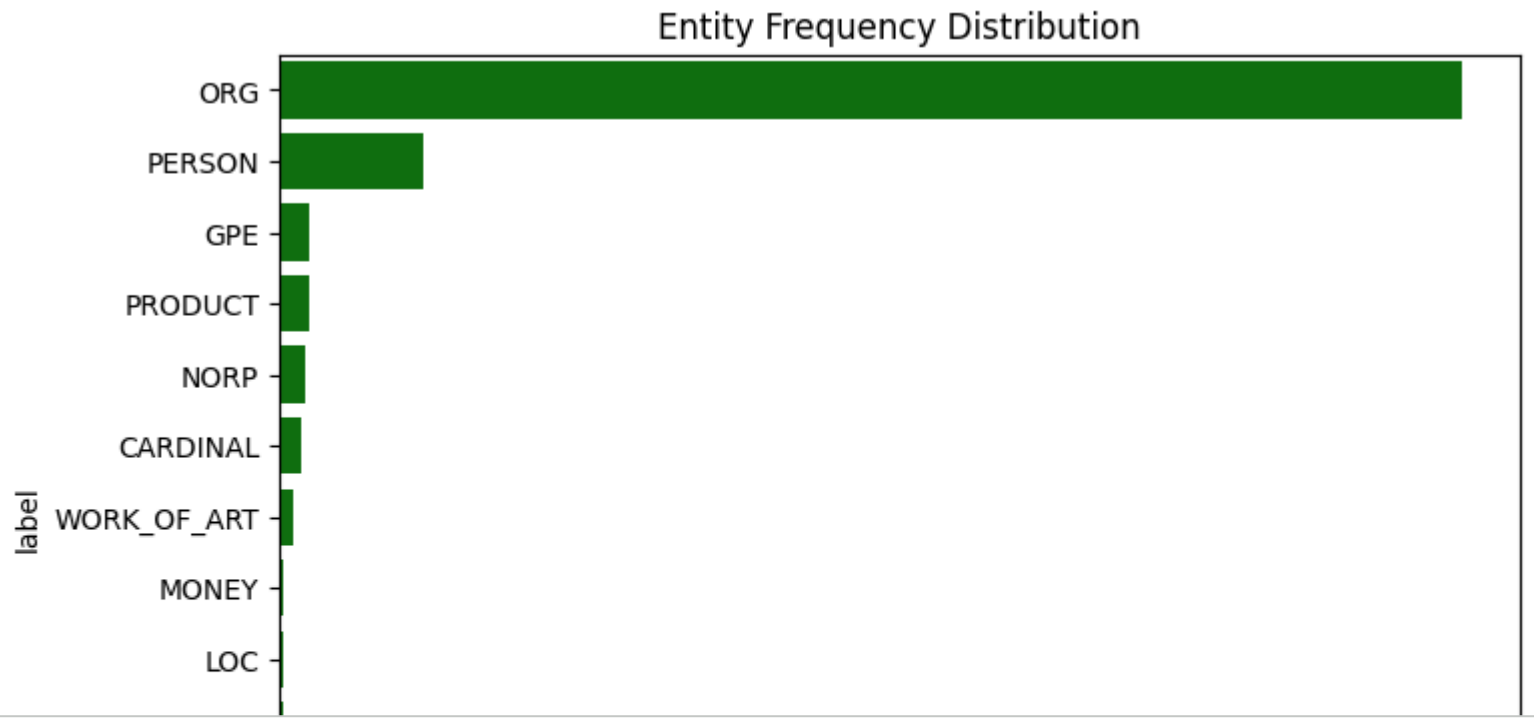
```
noun_phrases = []
for doc in docs:
    noun_phrases.extend([chunk.text for chunk in doc.noun_chunk
    np_df = pd.DataFrame(noun_phrases, columns=['phrase'])
top_np = np_df['phrase'].value_counts().head(15)
plt.figure(figsize=(10,6))
```

```
sns.barplot(x=top_np.values, y=top_np.index, color='blue')
plt.title("Top Noun Phrases in Abstracts")
plt.xlabel("Frequency")
plt.show()
```




```
entities = []
for doc in docs:
    for ent in doc.ents:
        entities.append((ent.text, ent.label_))

ent_df = pd.DataFrame(entities, columns=['entity','label'])
top_entities = ent_df['label'].value_counts()
plt.figure(figsize=(8,6))
sns.barplot(x=top_entities.values, y=top_entities.index, color="green")
plt.title("Entity Frequency Distribution")
plt.xlabel("Count")
plt.show()
```




```
matcher = Matcher(nlp.vocab)
pattern1 = [{"LOWER": "neural"}, {"LOWER": "network"}]
pattern2 = [{"LOWER": "support"}, {"LOWER": "vector"}, {"LOWER": "machine"}]

matcher.add("TECH_TERMS", [pattern1, pattern2])

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

pd.Series(matches).value_counts().head(10)
```

	count
Neural Network	17
neural network	8

```
essay_text = "Artificial intelligence is transforming research i
doc = nlp(essay_text)

print("Tokens:", [token.text for token in doc])
print("Noun Phrases:", [chunk.text for chunk in doc.noun_chunks]
print("Entities:", [(ent.text, ent.label_) for ent in doc.ents])
```

```
Tokens: ['Artificial', 'intelligence', 'is', 'transforming', 'research', 'in', 'computer', 'science', 'and', 'physics',
Noun Phrases: ['Artificial intelligence', 'research', 'computer science', 'physics']
Entities: []
```