Notebook Gemini

```
import pandas as pd

df = pd.read_csv('/content/arxiv_data.csv', engine='python')
display(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']

```
!pip install 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: 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) (2.1
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,!=1.8.1,<3.
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spac
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spac
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->spacy) (
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->sp
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packages (from typer-slim<1.0.0,>=0.3.0->spacy) (8
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2-
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->s
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<0.5.0,>=0
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-none-any
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 73.5 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 ast

# Define a safe literal_eval function to handle malformed strings or None values
def safe_literal_eval(s):
    try:
        if pd.isna(s): # Check for pandas NaN values, which can arise from None
            return []
        return ast.literal_eval(s)
    except (ValueError, SyntaxError): # Catch errors from malformed strings
        return []
```

```
# The 'terms' column contains strings that look like lists. We need to convert them to actual lists.
df['terms_list'] = df['terms'].apply(safe_literal_eval)

# Explode the lists to get individual terms and count their frequencies
all_terms = df['terms_list'].explode()

# Display the top 10 most frequent terms
print("Top 10 most frequent terms:")
display(all_terms.value_counts().head(10).to_frame())
```

```
Top 10 most frequent terms:
```

terms_list	count
cs.CV	3431
cs.LG	2403
stat.ML	1295
cs.AI	615
eess.IV	286
cs.NE	156
cs.CL	109
cs.SI	96
cs.RO	88
cs.GR	49

```
import spacy

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

```
# Select an abstract text from the DataFrame
abstract_text = df['summaries'].iloc[0]

# Process the text with the nlp pipeline
doc = nlp(abstract_text)

print(f"Processed abstract (first 200 chars):\n{str(doc)[:200]}...")
```

```
Processed abstract (first 200 chars):
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 appl...
```

```
print("Noun phrases:")
for chunk in doc.noun_chunks:
    print(chunk.text)
```

```
Noun phrases:
Stereo matching
the widely used techniques
depth
stereo images
its robustness
speed
It
the major
topics
research
it
its applications
autonomous driving
robotic navigation
3D reconstruction
many other fields
pixel
correspondences
non-textured, occluded and reflective areas
the major
challenge
stereo matching
Recent developments
that semantic cues
```

```
image segmentation
the results
stereo matching
Many deep neural network architectures
the
advantages
semantic segmentation
stereo matching
This paper
a comparison
the state
art networks
terms
accuracy
terms
speed
which
higher importance
real-time applications
```

```
print("Named entities:")
for ent in doc.ents:
    print(f"{ent.text} - {ent.label_}")
```

```
Named entities:
```

```
from spacy.matcher import Matcher

matcher = Matcher(nlp.vocab)

# Define a pattern for 'neural network'
pattern = [{"LOWER": "neural"}, {"LOWER": "network"}]
matcher.add("NEURAL_NETWORK", [pattern])

print("Matching 'neural network' pattern:")
matches = matcher(doc)
for match_id, start, end in matches:
    span = doc[start:end]
    print(span.text)
```

```
Matching 'neural network' pattern:
neural network
```

```
noun_phrases = []
for chunk in doc.noun_chunks:
    noun_phrases.append(chunk.text.lower())

noun_phrase_series = pd.Series(noun_phrases)
top_10_noun_phrases = noun_phrase_series.value_counts().head(10)

print("Top 10 most frequent noun phrases:")
display(top_10_noun_phrases.to_frame())
```

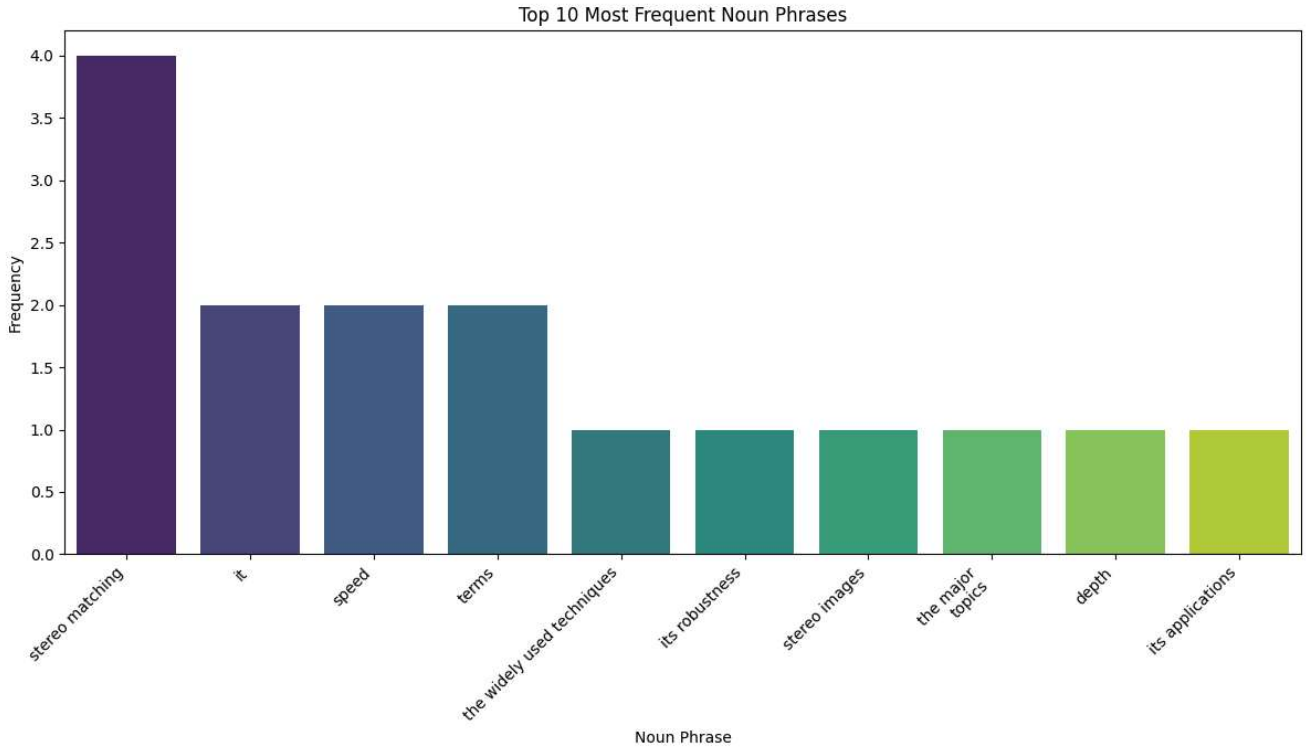
```
Top 10 most frequent noun phrases:
```

	count
stereo matching	4
it	2
speed	2
terms	2
the widely used techniques	1
its robustness	1
stereo images	1
the major\ntopics	1
depth	1
its applications	1

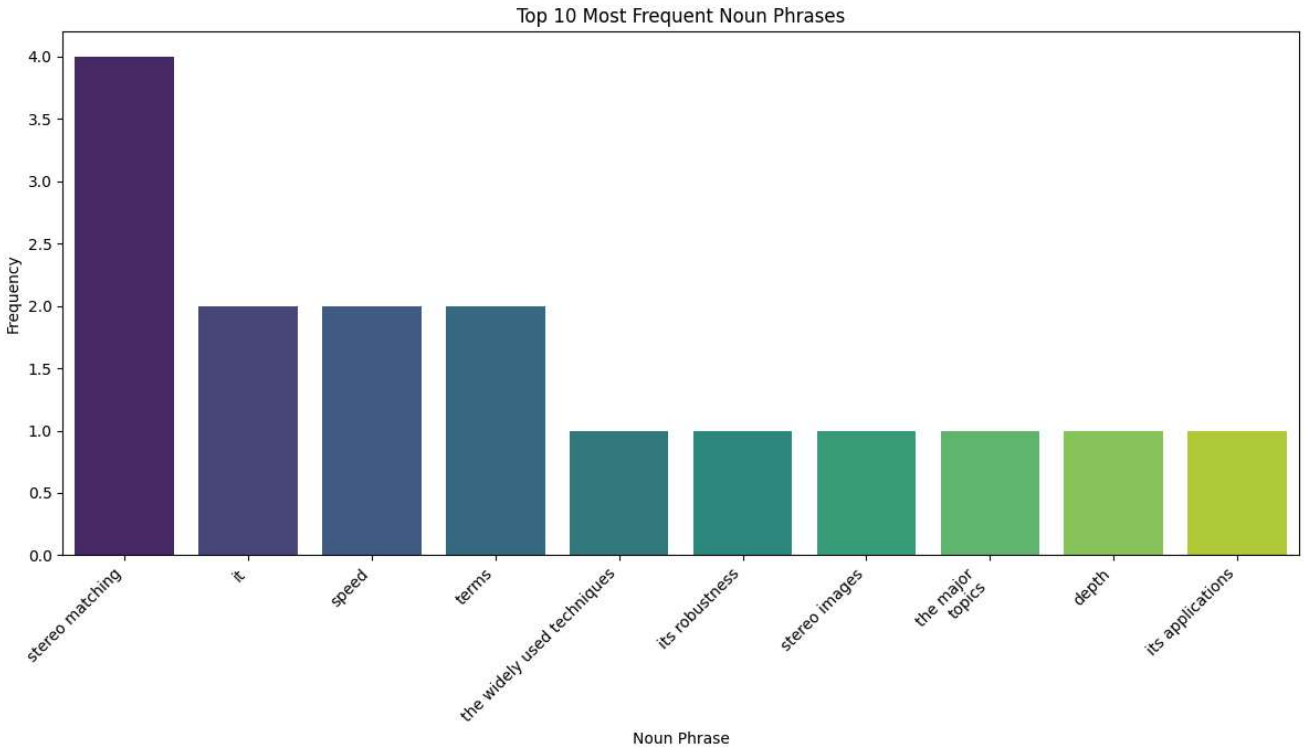
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 7))
```

```
sns.barplot(x=top_10_noun_phrases.index, y=top_10_noun_phrases.values, palette='viridis')
plt.title('Top 10 Most Frequent Noun Phrases')
plt.xlabel('Noun Phrase')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-4162454024.py:5: FutureWarning:

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

  sns.barplot(x=top_10_noun_phrases.index, y=top_10_noun_phrases.values, palette='viridis')
```




```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 7))
sns.barplot(x=top_10_noun_phrases.index, y=top_10_noun_phrases.values, hue=top_10_noun_phrases.index, palette='viridis', legend
plt.title('Top 10 Most Frequent Noun Phrases')
plt.xlabel('Noun Phrase')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```




```
named_entities = []
for ent in doc.ents:
    named_entities.append(ent.text.lower())

named_entity_series = pd.Series(named_entities)
named_entity_counts = named_entity_series.value_counts()

print("Named Entity Frequencies:")
display(named_entity_counts.to_frame())
```

```
Named Entity Frequencies:
```

No entries Filter

index	count

Show 10 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

```
import matplotlib.pyplot as plt
import seaborn as sns

if not named_entity_counts.empty:
    plt.figure(figsize=(12, 7))
    sns.barplot(x=named_entity_counts.index, y=named_entity_counts.values, hue=named_entity_counts.index, palette='viridis', le
    plt.title('Named Entity Frequencies')
    plt.xlabel('Named Entity')
    plt.ylabel('Frequency')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()
else:
    print("No named entities found in the text, thus no plot generated.")
```

```
No named entities found in the text, thus no plot generated.
```