Text Data visualizations : Sentiment Analysis, Name Entitiy , Sentiment Variation over the story

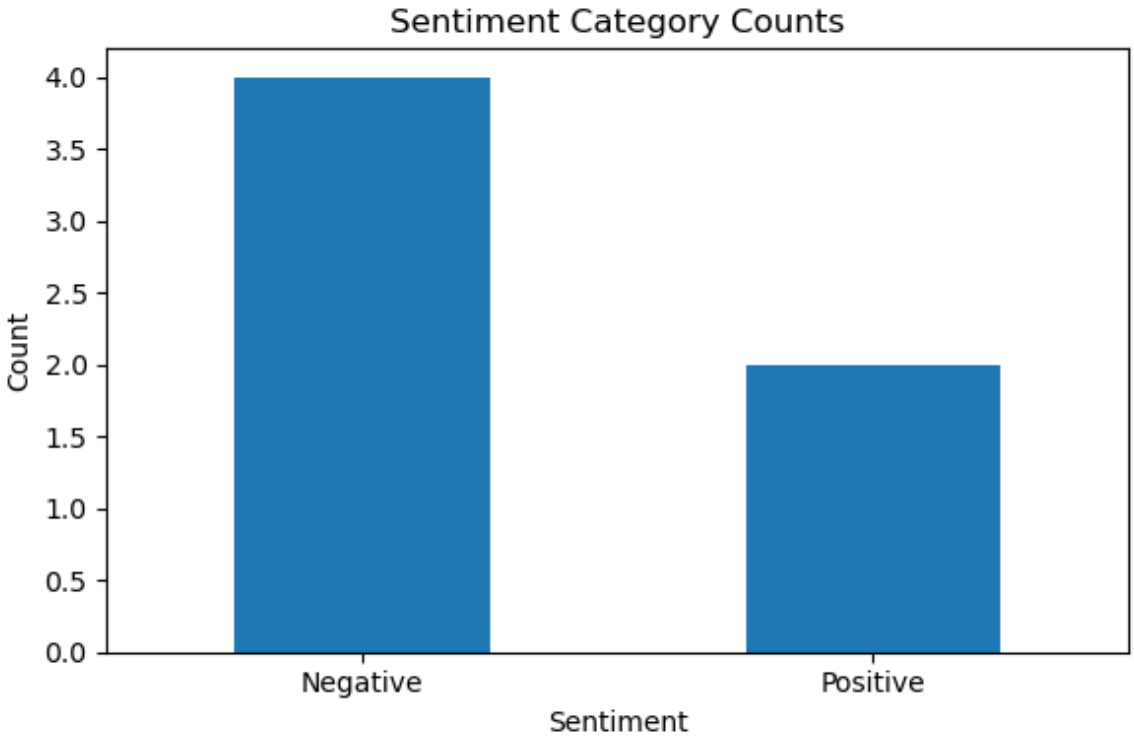
pip install spacy

In [4]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
import pandas as pd
import nltk

# ---------------------------
# 1. Sample Text Data
# ---------------------------
texts = [
    "I love this movie! It's fantastic.",
    "This is the worst product I ever bought.",
    "The experience was okay, nothing special.",
    "I absolutely adore the design of this phone.",
    "The service was terrible, I'm disappointed.",
    "It's fine. Not good, not bad.",
]

# ---------------------------
# 2. Initialize VADER
# ---------------------------
nltk.download('vader_lexicon')
sia = SentimentIntensityAnalyzer()

# ---------------------------
# 3. Compute Sentiment Scores
# ---------------------------
scores = []
labels = []

for t in texts:
    s = sia.polarity_scores(t)
    scores.append(s['compound'])

    # classify based on compound score
    if s['compound'] >= 0.05:
        labels.append("Positive")
    elif s['compound'] <= -0.05:
        labels.append("Negative")
    else:
        labels.append("Neutral")

# Create DataFrame for easy plotting
df = pd.DataFrame({"text": texts, "compound": scores, "label": labe

print(df)
```

```
# ----------------------------
# 4. Count sentiment classes
# ----------------------------
counts = df['label'].value_counts()

# ----------------------------
# 5. Bar Chart
# ----------------------------
plt.figure(figsize=(6, 4))
counts.plot(kind='bar')
plt.title("Sentiment Category Counts")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

	text	compound	label
0	I love this movie! It's fantastic.	0.8439	Positive
1	This is the worst product I ever bought.	-0.6249	Negative
2	The experience was okay, nothing special.	-0.0920	Negative
3	I absolutely adore the design of this phone.	0.5984	Positive
4	The service was terrible, I'm disappointed.	-0.7351	Negative
5	It's fine. Not good, not bad.	-0.4543	Negative

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/madhukumari/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```



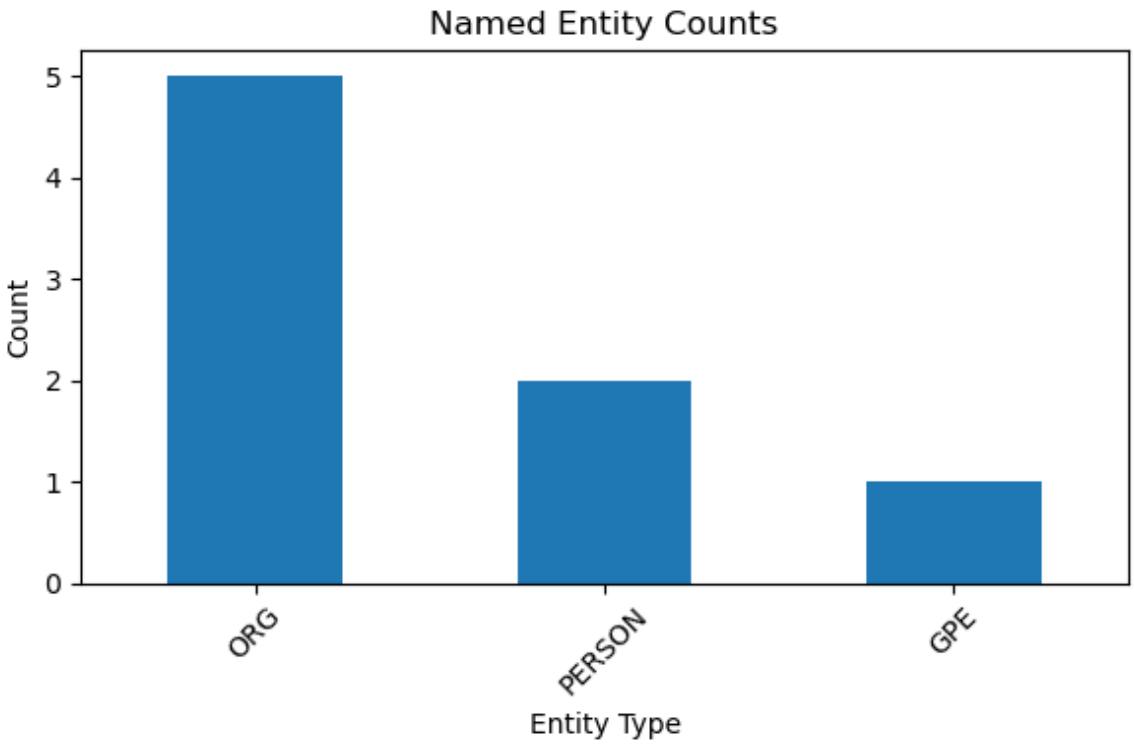
!pip install spacy matplotlib pandas !python -m spacy download en_core_web_sm

Name Entity Recognition (NER)

In [5]:
```
import spacy
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Load model
nlp = spacy.load("en_core_web_sm")

texts = [
    "Elon Musk founded SpaceX and leads Tesla.",
    "Bill Gates created Microsoft and works with the WHO.",
    "Google opened a new campus in Toronto.",
    "Pfizer partnered with BioNTech to develop a COVID vaccine.",
]

# Extract entities
all_ents = []
for t in texts:
    doc = nlp(t)
    for ent in doc.ents:
        all_ents.append((ent.text, ent.label_))

df = pd.DataFrame(all_ents, columns=["entity", "label"])
print(df)

# Counts per entity type
label_counts = df["label"].value_counts()

# Plot
plt.figure(figsize=(6,4))
label_counts.plot(kind="bar")
plt.title("Named Entity Counts")
plt.xlabel("Entity Type")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

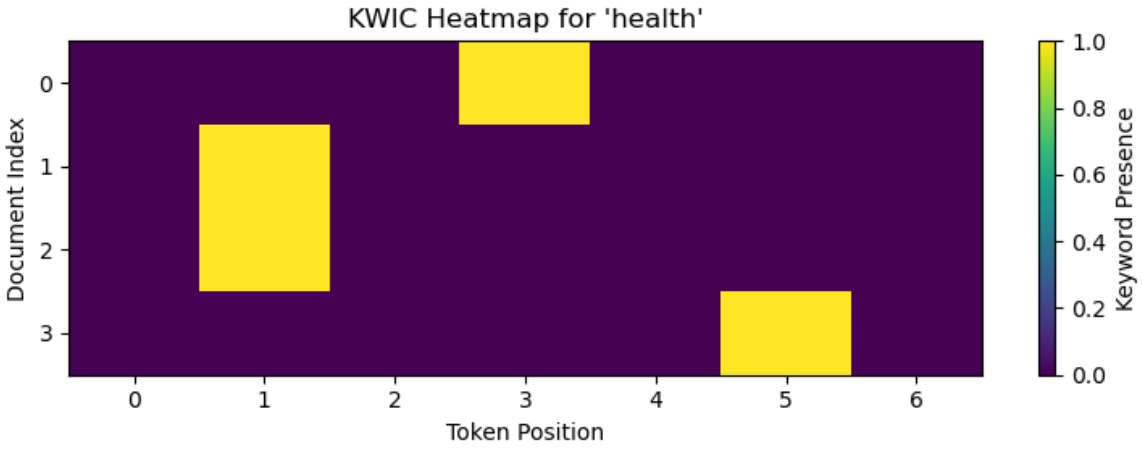
```
      entity   label
0   Elon Musk  PERSON
1       Tesla     ORG
2  Bill Gates  PERSON
3   Microsoft     ORG
4         WHO     ORG
5      Google     ORG
6     Toronto     GPE
7       COVID     ORG
```



NER + Sentiment Visualization

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt

docs = [
    "Machine learning improves health systems worldwide.",
    "Public health initiatives are expanding rapidly.",
    "Mental health is gaining global attention.",
    "Economic policies indirectly shape population health outcomes."
]

keyword = "health"

# Tokenize
tokenized = [d.lower().replace(".", "").split() for d in docs]
max_len = max(len(t) for t in tokenized)
```

```
# KWIC matrix
kwic = np.zeros((len(docs), max_len))

for i, tokens in enumerate(tokenized):
    for j, tok in enumerate(tokens):
        if tok == keyword:
            kwic[i][j] = 1

# Heatmap
plt.figure(figsize=(8,3))
plt.imshow(kwic, aspect="auto", interpolation="nearest")
plt.title(f"KWIC Heatmap for '{keyword}'")
plt.xlabel("Token Position")
plt.ylabel("Document Index")
plt.colorbar(label="Keyword Presence")
plt.tight_layout()
plt.show()

# Print KWIC context
print("\nKWIC contexts:")
window = 3
for i, tokens in enumerate(tokenized):
    for j, tok in enumerate(tokens):
        if tok == keyword:
            start = max(0, j-window)
            end = min(len(tokens), j+window+1)
            context = " ".join(tokens[start:end])
            print(f"Doc {i}, position {j}: ... {context} ...")
```



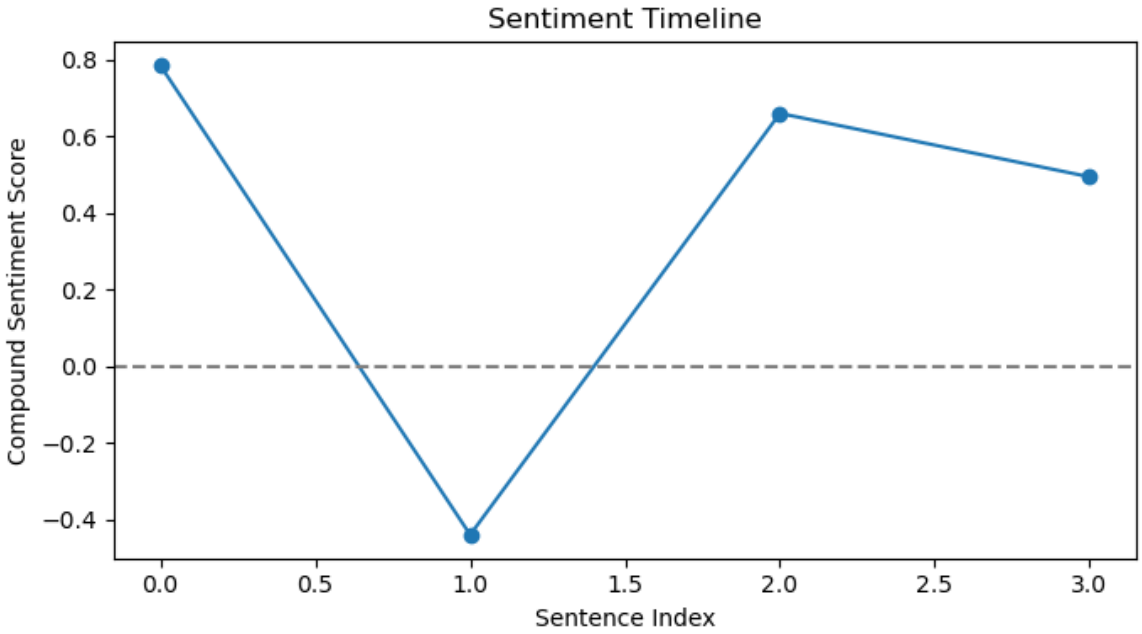

```
KWIC contexts:
Doc 0, position 3: ... machine learning improves health systems worl
dwide ...
Doc 1, position 1: ... public health initiatives are expanding ...
Doc 2, position 1: ... mental health is gaining global ...
Doc 3, position 5: ... indirectly shape population health outcomes
...
```

Sentiment variation over the story

In [7]:

```
from nltk.sentiment import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
```

```
sia = SentimentIntensityAnalyzer()

text_story = [
    "The day started beautifully with sunshine.",
    "Later, heavy traffic made the commute frustrating.",
    "At work, a surprise appreciation email lifted my mood.",
    "The evening ended with a relaxing walk in the park."
]

# Compute sentiment for each sentence
compound_scores = [sia.polarity_scores(sent)["compound"] for sent i

# Plot timeline
plt.figure(figsize=(7,4))
plt.plot(range(len(text_story)), compound_scores, marker="o")
plt.title("Sentiment Timeline")
plt.xlabel("Sentence Index")
plt.ylabel("Compound Sentiment Score")
plt.axhline(0, color="gray", linestyle="--")
plt.tight_layout()
plt.show()

print(list(zip(text_story, compound_scores)))
```




```
[('The day started beautifully with sunshine.', 0.7845), ('Later, he
avy traffic made the commute frustrating.', -0.4404), ('At work, a s
urprise appreciation email lifted my mood.', 0.6597), ('The evening
ended with a relaxing walk in the park.', 0.4939)]
```

Average Sentiment Per Entity

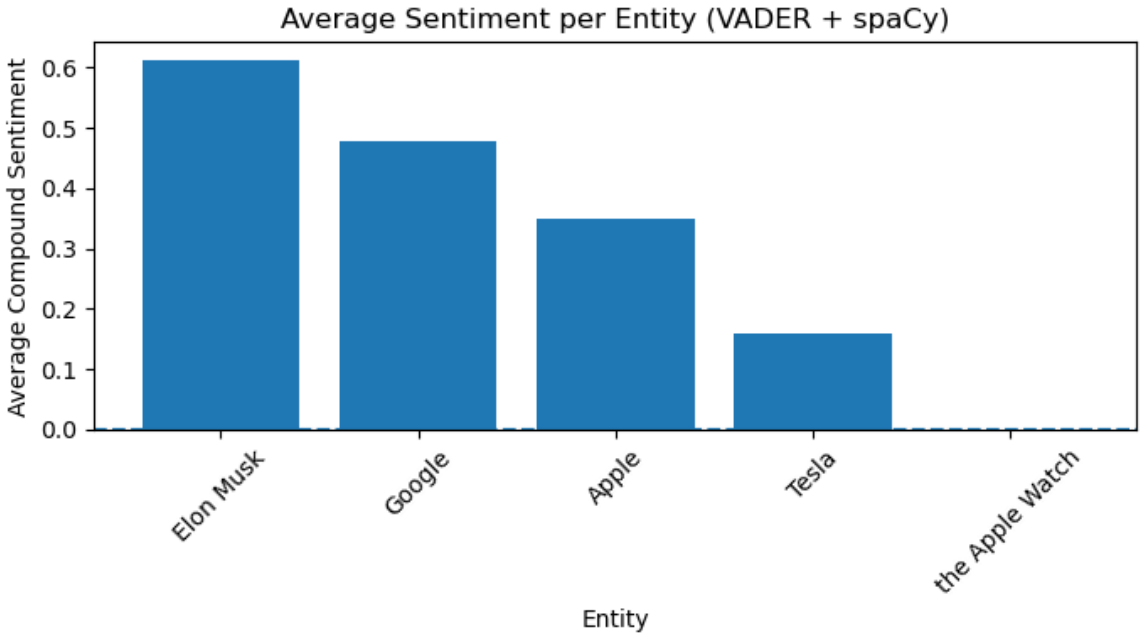
In [ ]:

In [8]:
```
import spacy
from nltk.sentiment import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
import pandas as pd
```

```
from collections import defaultdict

# ----------------------------
# 1. Load models
# ----------------------------
nlp = spacy.load("en_core_web_sm")
sia = SentimentIntensityAnalyzer()

# ----------------------------
# 2. Sample text data
#    (You can replace this with your own paragraphs/articles)
# ----------------------------
texts = [
    "Elon Musk announced a new Tesla model and investors seemed very
    "Some critics said that Tesla's quality has declined recently."
    "Meanwhile, Apple released new health features on the Apple Wat
    "Analysts believe Apple will benefit strongly from the growing
    "Google improved its policies for privacy concerns.",
]

# ----------------------------
# 3. Process texts sentence by sentence
# ----------------------------
entity_sentiments = defaultdict(list)

for doc_text in texts:
    doc = nlp(doc_text)
    for sent in doc.sents:
        sent_text = sent.text
        # Sentiment for the sentence
        score = sia.polarity_scores(sent_text)["compound"]

        # NER on the sentence
        sent_doc = nlp(sent_text)
        for ent in sent_doc.ents:
            # Filter to some common entity types (you can adjust)
            if ent.label_ in {"PERSON", "ORG", "GPE"}:
                key = ent.text.strip()
                entity_sentiments[key].append(score)

# ----------------------------
# 4. Aggregate sentiment per entity (mean)
# ----------------------------
data = []
for ent, scores in entity_sentiments.items():
    avg_score = sum(scores) / len(scores)
    data.append((ent, avg_score, len(scores)))

df = pd.DataFrame(data, columns=["entity", "avg_sentiment", "mentio
print("Entity-level sentiment summary:")
print(df)

# ----------------------------
# 5. Plot average sentiment per entity
# ----------------------------
df_sorted = df.sort_values("avg_sentiment", ascending=False)
```

```
plt.figure(figsize=(7, 4))
plt.bar(df_sorted["entity"], df_sorted["avg_sentiment"])
plt.title("Average Sentiment per Entity (VADER + spaCy)")
plt.xlabel("Entity")
plt.ylabel("Average Compound Sentiment")
plt.xticks(rotation=45)
plt.axhline(0, linestyle="--")
plt.tight_layout()
plt.show()
```

```
Entity-level sentiment summary:
            entity  avg_sentiment  mention_count
0        Elon Musk        0.61150              1
1            Tesla        0.15775              2
2            Apple        0.35015              2
3  the Apple Watch        0.00000              1
4           Google        0.47670              1
```



In [9]:
```
# Text Segmentwise sentiment over the text
```

In [ ]:
```

```

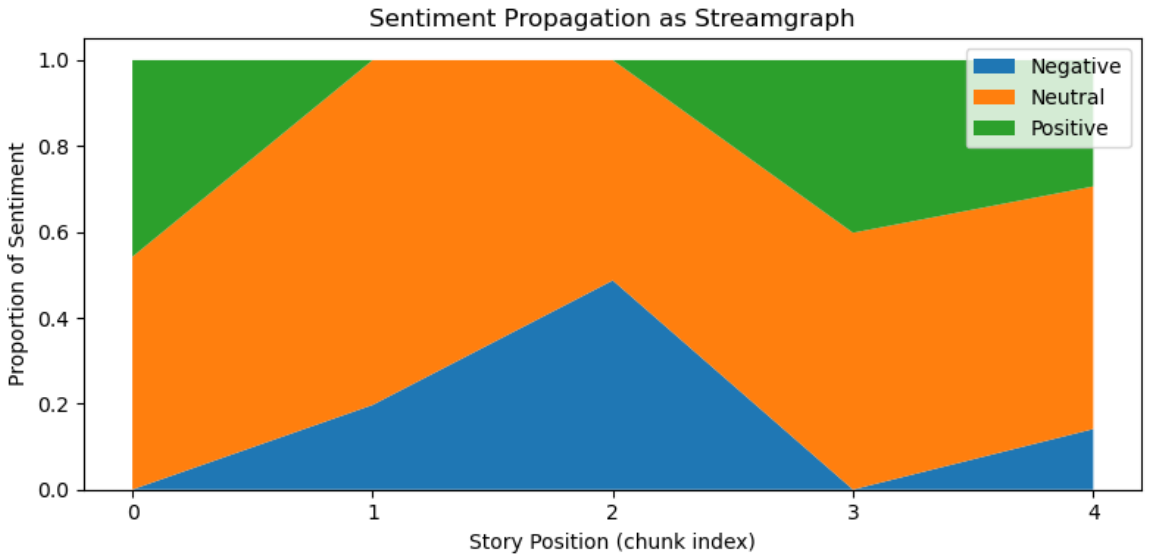
In [ ]:
```

```

In [ ]:
```

```

In [10]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
import numpy as np

# ----------------------------
# 1. Sample "story" as a sequence of chunks
#    (These could be sentences, paragraphs, time windows, etc.)
# ----------------------------
story_chunks = [
    "The morning was bright and full of hope. Birds were singing an
```

```
    "By midday, clouds gathered and small problems started to appea
    "In the afternoon, a major system crash caused frustration and 
    "Later, the issue was resolved and everyone felt relieved and p
    "The day ended quietly, with a calm sense of satisfaction and a
]

# ----------------------------
# 2. Initialize VADER
# ----------------------------
sia = SentimentIntensityAnalyzer()

# ----------------------------
# 3. Compute sentiment for each chunk
#    We'll use the pos/neu/neg scores directly
# ----------------------------
pos_scores = []
neu_scores = []
neg_scores = []

for chunk in story_chunks:
    scores = sia.polarity_scores(chunk)
    pos_scores.append(scores["pos"])
    neu_scores.append(scores["neu"])
    neg_scores.append(scores["neg"])

# Convert to numpy arrays (optional, but handy)
pos_scores = np.array(pos_scores)
neu_scores = np.array(neu_scores)
neg_scores = np.array(neg_scores)

# Normalization (optional): ensure each time step sums to 1
total = pos_scores + neu_scores + neg_scores
pos_norm = pos_scores / total
neu_norm = neu_scores / total
neg_norm = neg_scores / total

# ----------------------------
# 4. Create a streamgraph (stacked area chart)
# ----------------------------
x = np.arange(len(story_chunks))  # time axis: chunk indices

plt.figure(figsize=(8, 4))

# stackplot makes a nice stream-like visualization
plt.stackplot(x, neg_norm, neu_norm, pos_norm, labels=["Negative", '

plt.title("Sentiment Propagation as Streamgraph")
plt.xlabel("Story Position (chunk index)")
plt.ylabel("Proportion of Sentiment")
plt.legend(loc="upper right")
plt.xticks(x, [f"{i}" for i in x])  # label positions 0,1,2,...
plt.tight_layout()
plt.show()

# ----------------------------
# 5. (Optional) Print numeric values for reference
```

```
# ----------------------------
for i, chunk in enumerate(story_chunks):
    print(f"\nChunk {i}:")
    print("Text:", chunk)
    print(f"Negative: {neg_norm[i]:.2f}, Neutral: {neu_norm[i]:.2f}
```




```
Chunk 0:
Text: The morning was bright and full of hope. Birds were singing an
d people smiled.
Negative: 0.00, Neutral: 0.54, Positive: 0.46

Chunk 1:
Text: By midday, clouds gathered and small problems started to appea
r at work.
Negative: 0.20, Neutral: 0.80, Positive: 0.00

Chunk 2:
Text: In the afternoon, a major system crash caused frustration and
anger among the team.
Negative: 0.49, Neutral: 0.51, Positive: 0.00

Chunk 3:
Text: Later, the issue was resolved and everyone felt relieved and p
roud of their effort.
Negative: 0.00, Neutral: 0.60, Positive: 0.40

Chunk 4:
Text: The day ended quietly, with a calm sense of satisfaction and a
bit of exhaustion.
Negative: 0.14, Neutral: 0.56, Positive: 0.29
```

In [ ]:

In [ ]: