Dependency_Parsing_Assignment (1)

March 5, 2025

0.1 Introduction

Twitter has become a prominent platform for expressing opinions, emotions, and sentiments. However, the short, informal, and often noisy nature of tweets presents persistent challenges for automatic sentiment analysis (Poria et al., 2020). Prior research has largely focused on lexical and semantic features—such as word embeddings, sentiment lexicons, and n-grams—to classify sentiment in social media texts (Rosenthal et al., 2019; Barbieri et al., 2020). While these approaches have driven advances in performance, they may overlook deeper linguistic properties of text, such as syntactic structures, which can carry crucial contextual or affective information.

Studies have begun to suggest that linguistic features, including syntactic complexity, part-of-speech patterns, and dependency relations, can provide valuable signals in sentiment analysis, especially in contexts where sentiment is expressed indirectly or through complex constructions (Ghosh et al., 2015; Zhang et al., 2018). For instance, sarcastic or negative sentiment may rely more on subordinate clauses, contrastive structures, or embedded phrases, which are not captured purely through word-level features.

Despite these insights, the role of syntactic complexity in tweet sentiment classification remains underexplored. The TweetEval benchmark (Barbieri et al., 2020) provides an ideal testbed to investigate this, offering a large, labeled dataset of tweets categorized as positive, neutral, or negative. Using dependency parsing with spaCy, this study examines whether measures of clausal and phrasal complexity differ systematically across sentiment categories. Understanding these patterns could inform the development of more linguistically informed sentiment classifiers and shed light on how structural features contribute to sentiment expression in social media text. ## **Research Question**

"Can measures of clausal and phrasal complexity help distinguish between sentiment categories (positive, neutral, negative) in tweets?"

0.2 Data

I use the TweetEval Sentiment Analysis dataset, which includes tweets labeled as positive, neutral, or negative. This dataset is ideal for examining syntactic complexity in relation to sentiment, as it contains a large, annotated collection of real-world tweets, characterized by diverse linguistic styles.

Syntactic Feature Extraction Using spaCy's dependency parser, we extract the following syntactic complexity features for each tweet:

Clausal Complexity:

- Subordinate clause count (mark, advcl, ccomp, xcomp).

- Conjunction count (cc).
- Sentence length (number of tokens). Phrasal Complexity:
- Average noun phrase length.
- Prepositional phrase count (prep).
- Adjective modifier count (amod).

These features are chosen based on prior studies that link complex syntactic structures with nuanced or indirect sentiment expressions (Ghosh et al., 2015).

Dataset Summary: - Dataset Split Number of Tweets - Training 45,389 - Validation 2,000 - Test 11,906

Each tweet is preprocessed to clean non-textual elements (such as URLs and mentions), and only the core textual content is analyzed for syntactic features. Dependency parsing is performed using spaCy, ensuring consistent and accurate extraction of linguistic structures.

Here I load the dataset and see the info:

[15]:
```
! pip install datasets
```

```
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-
packages (3.3.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.17.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-
packages (from datasets) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages
(from datasets) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2024.12.0,>=2023.1.0 in
/usr/local/lib/python3.11/dist-packages (from
fsspec[http]<=2024.12.0,>=2023.1.0->datasets) (2024.10.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.11.13)
Requirement already satisfied: huggingface-hub>=0.24.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.28.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
```

```
packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.4.6)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.0->datasets)
(4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
```

[16]:
```
from datasets import load_dataset

ds = load_dataset("cardiffnlp/tweet_eval", "sentiment")
```

[17]:
```
print(ds)
from collections import Counter
```

```
Counter(ds["train"]["label"])
```

```
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 45615
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 12284
    })
    validation: Dataset({
        features: ['text', 'label'],
        num_rows: 2000
    })
})
```

```
[17]: Counter({2: 17849, 1: 20673, 0: 7093})
```

Here is the example of this dataset.

```
[18]: for i in range(5):
          print(ds["train"][i])
```

```
{'text': '"QT @user In the original draft of the 7th book, Remus Lupin survived
the Battle of Hogwarts. #HappyBirthdayRemusLupin"', 'label': 2}
{'text': '"Ben Smith / Smith (concussion) remains out of the lineup Thursday,
Curtis #NHL #SJ"', 'label': 1}
{'text': 'Sorry bout the stream last night I crashed out but will be on tonight
for sure. Then back to Minecraft in pc tomorrow night.', 'label': 1}
{'text': "Chase Headley's RBI double in the 8th inning off David Price snapped a
Yankees streak of 33 consecutive scoreless innings against Blue Jays", 'label':
1}
{'text': '@user Alciato: Bee will invest 150 million in January, another 200 in
the Summer and plans to bring Messi by 2017"', 'label': 2}
```

0.3 Analysis

The primary goal of this analysis is to investigate whether syntactic complexity, measured through clausal and phrasal features, differs across positive, neutral, and negative tweets. The analysis consists of the following steps:

1.Feature Extraction Using spaCy's dependency parser, I extract syntactic complexity features from each tweet in the dataset.

```
[19]: # Feature extraction
      import spacy
      import pandas as pd
      from datasets import load_dataset
```

```python
from tqdm import tqdm

# Load dataset
ds = load_dataset("cardiffnlp/tweet_eval", "sentiment")
df = pd.DataFrame(ds["train"])

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

# Define feature extraction function
def extract_syntactic_features(text):
    doc = nlp(text)

    # Clausal complexity
    subordinate_clauses = sum(1 for token in doc if token.dep_ in ['mark',␣
 ↪'advcl', 'ccomp', 'xcomp'])
    conjunctions = sum(1 for token in doc if token.dep_ == 'cc')
    sentence_length = len(doc)

    # Phrasal complexity
    noun_phrase_lengths = [len(np) for np in doc.noun_chunks]
    avg_noun_phrase_length = sum(noun_phrase_lengths) /␣
 ↪len(noun_phrase_lengths) if noun_phrase_lengths else 0
    prepositional_phrases = sum(1 for token in doc if token.dep_ == 'prep')
    adjective_modifiers = sum(1 for token in doc if token.dep_ == 'amod')

    return {
        'subordinate_clauses': subordinate_clauses,
        'conjunctions': conjunctions,
        'sentence_length': sentence_length,
        'avg_noun_phrase_length': avg_noun_phrase_length,
        'prepositional_phrases': prepositional_phrases,
        'adjective_modifiers': adjective_modifiers
    }

# Apply feature extraction
features = []
for tweet in tqdm(df["text"]):
    features.append(extract_syntactic_features(tweet))

# Add features to DataFrame
features_df = pd.DataFrame(features)
df = pd.concat([df, features_df], axis=1)

# Map numerical labels to text
df['sentiment'] = df['label'].map({0: 'negative', 1: 'neutral', 2: 'positive'})
```

```
# Preview the dataset with features
df.head()
```

```
100%|          | 45615/45615 [05:59<00:00, 126.91it/s]
```

[19]:
```
                                                text  label  \
0  "QT @user In the original draft of the 7th boo...      2
1  "Ben Smith / Smith (concussion) remains out of...      1
2  Sorry bout the stream last night I crashed out...      1
3  Chase Headley's RBI double in the 8th inning o...      1
4  @user Alciato: Bee will invest 150 million in ...      2

   subordinate_clauses  conjunctions  sentence_length  avg_noun_phrase_length  \
0                    0             0               23                2.200000
1                    0             0               21                2.000000
2                    0             1               26                1.333333
3                    0             0               24                3.000000
4                    1             1               24                1.250000

   prepositional_phrases  adjective_modifiers sentiment
0                      3                    2  positive
1                      2                    0   neutral
2                      4                    2   neutral
3                      3                    2   neutral
4                      3                    0  positive
```

First calculate descriptive statistics to observe how each syntactic feature varies across sentiment categories.

[20]:
```
feature_columns = [
    'subordinate_clauses',
    'conjunctions',
    'sentence_length',
    'avg_noun_phrase_length',
    'prepositional_phrases',
    'adjective_modifiers'
]

for feature in feature_columns:
    print(f"\n--- {feature} ---")
    print(df.groupby('sentiment')[feature].describe())
```

```

--- subordinate_clauses ---
             count      mean       std  min  25%  50%  75%  max
sentiment
negative    7093.0  1.426336  1.381366  0.0  0.0  1.0  2.0  9.0
neutral    20673.0  1.085280  1.259074  0.0  0.0  1.0  2.0  8.0
```

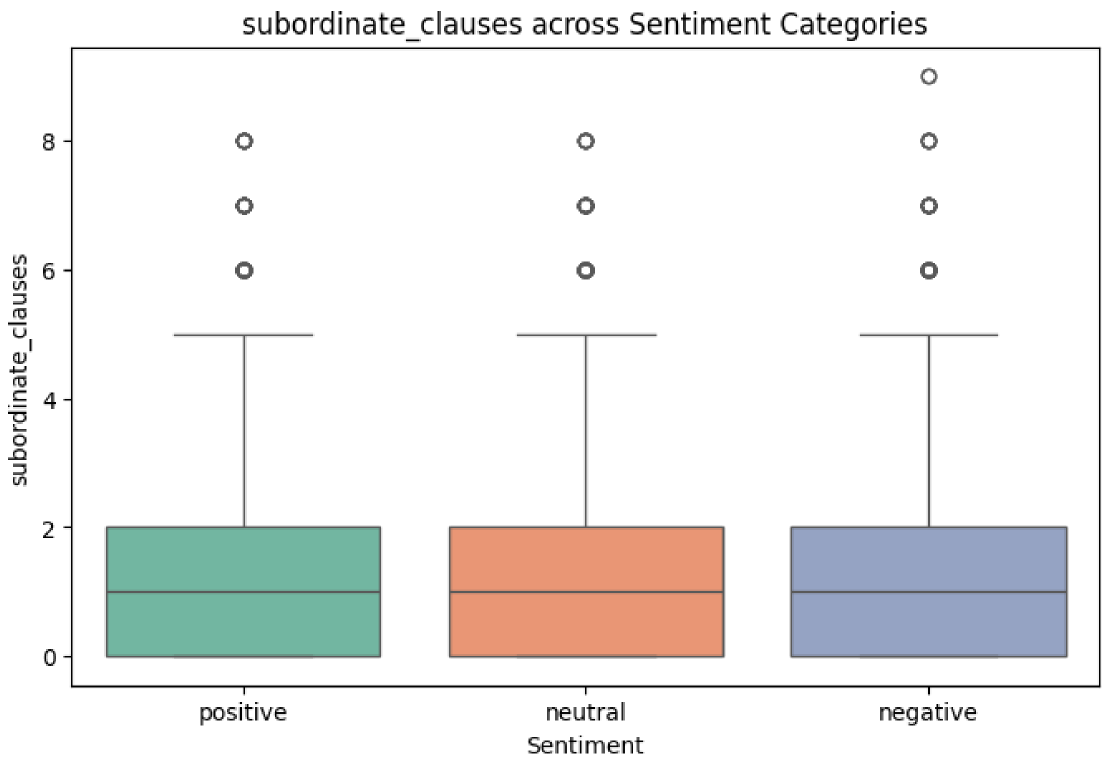
```
positive   17849.0  1.068743  1.217602  0.0  0.0  1.0  2.0  8.0

--- conjunctions ---
             count      mean       std  min  25%  50%  75%  max
sentiment
negative    7093.0  0.513464  0.685023  0.0  0.0  0.0  1.0  5.0
neutral    20673.0  0.460262  0.670647  0.0  0.0  0.0  1.0  5.0
positive   17849.0  0.476553  0.672940  0.0  0.0  0.0  1.0  9.0

--- sentence_length ---
             count       mean       std  min   25%   50%   75%   max
sentiment
negative    7093.0  24.078669  6.039540  4.0  20.0  25.0  28.0  48.0
neutral    20673.0  22.765153  6.415339  2.0  18.0  23.0  27.0  67.0
positive   17849.0  23.131716  6.351100  4.0  19.0  23.0  28.0  55.0

--- avg_noun_phrase_length ---
             count      mean       std  min  25%       50%   75%   max
sentiment
negative    7093.0  1.830005  0.561514  0.0  1.5  1.750000  2.00  11.0
neutral    20673.0  1.972843  0.667061  0.0  1.5  1.833333  2.25  13.0
positive   17849.0  1.928428  0.629676  0.0  1.5  1.800000  2.20  15.0

--- prepositional_phrases ---
             count      mean       std  min  25%  50%  75%  max
sentiment
negative    7093.0  1.830678  1.241420  0.0  1.0  2.0  3.0  8.0
neutral    20673.0  2.025250  1.299202  0.0  1.0  2.0  3.0  8.0
positive   17849.0  1.900835  1.254585  0.0  1.0  2.0  3.0  8.0

--- adjective_modifiers ---
             count      mean       std  min  25%  50%  75%  max
sentiment
negative    7093.0  0.854363  0.952670  0.0  0.0  1.0  1.0  9.0
neutral    20673.0  0.741112  0.889242  0.0  0.0  1.0  1.0  7.0
positive   17849.0  0.884531  0.944831  0.0  0.0  1.0  1.0  7.0
```

Visualization

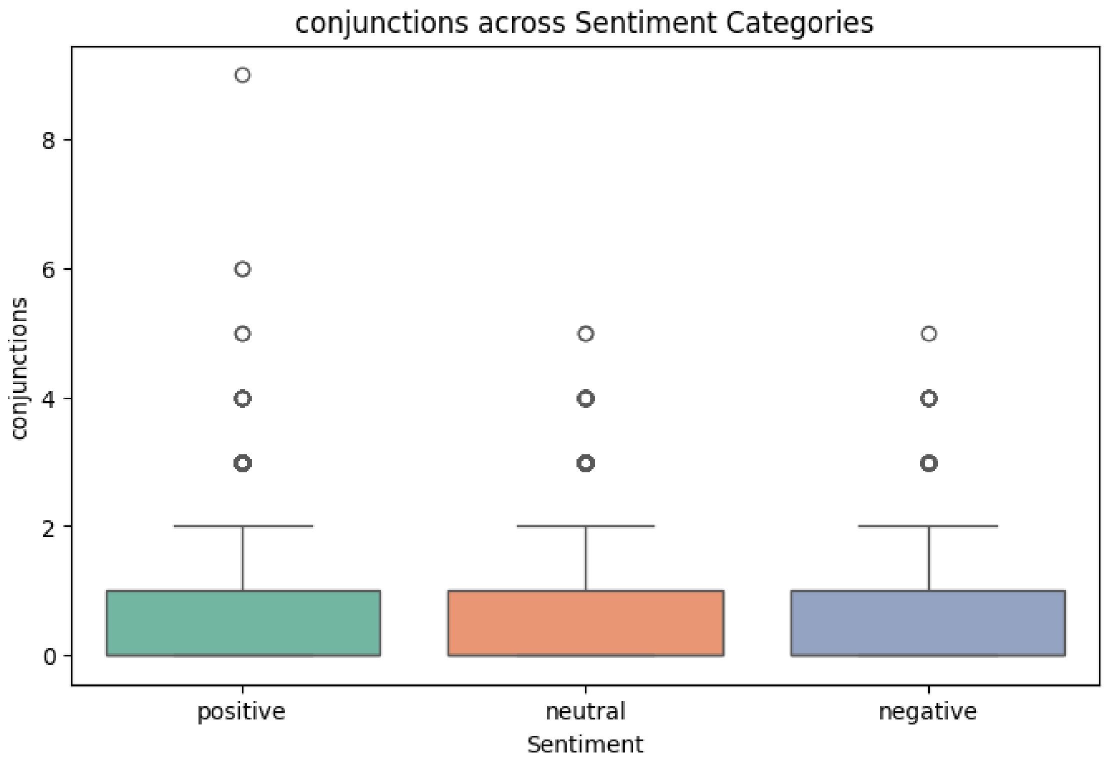
I use boxplots to visually compare each feature across the three sentiment categories.

[21]:
```
import seaborn as sns
import matplotlib.pyplot as plt

for feature in feature_columns:
    plt.figure(figsize=(8, 5))
    sns.boxplot(x='sentiment', y=feature, data=df, palette="Set2")
    plt.title(f'{feature} across Sentiment Categories')
    plt.xlabel('Sentiment')
    plt.ylabel(feature)
    plt.show()
```

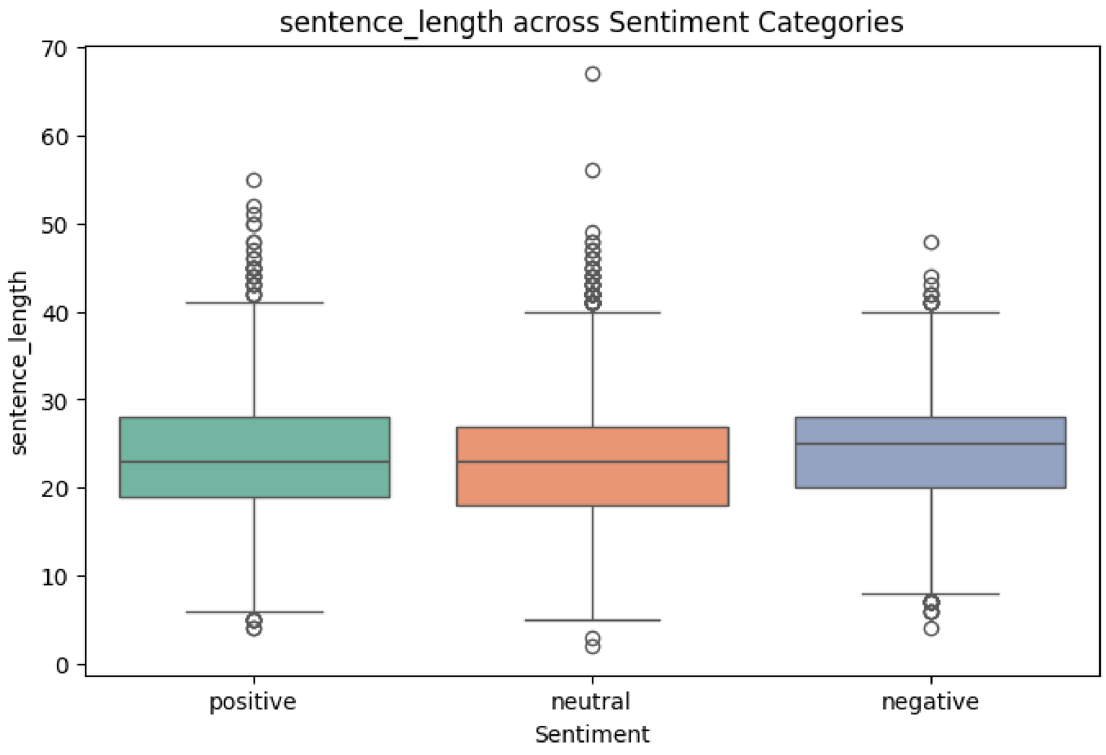
```
<ipython-input-21-cfee21983038>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='sentiment', y=feature, data=df, palette="Set2")
```



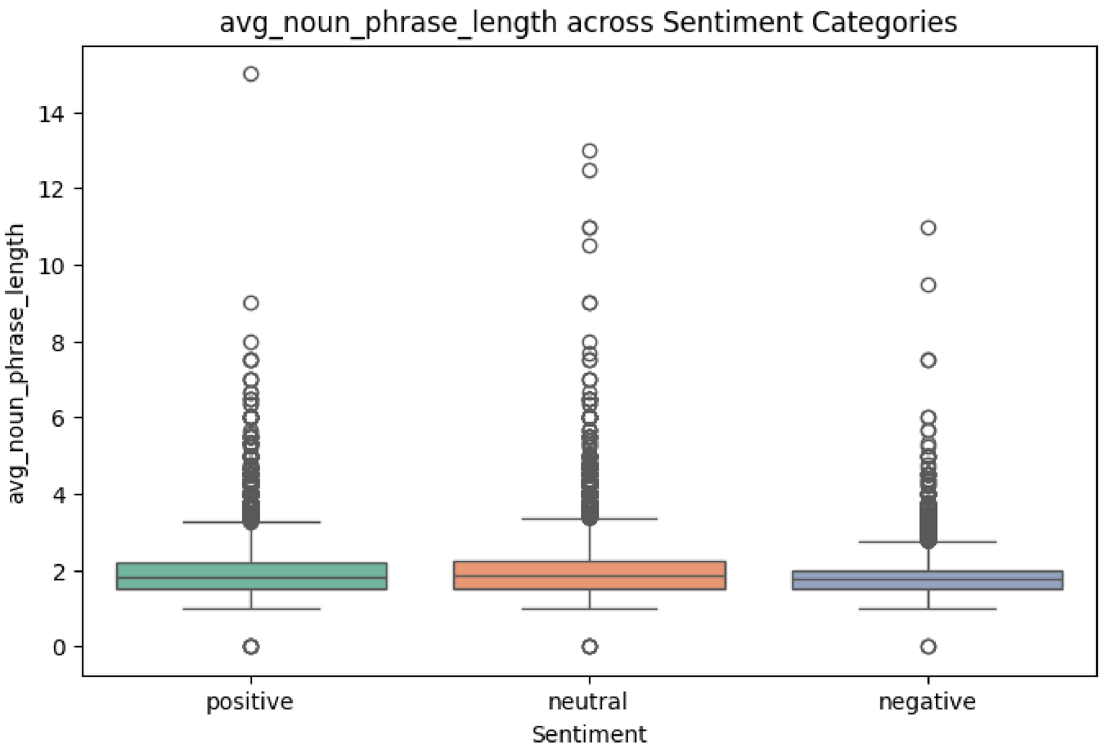
```
<ipython-input-21-cfee21983038>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='sentiment', y=feature, data=df, palette="Set2")
```



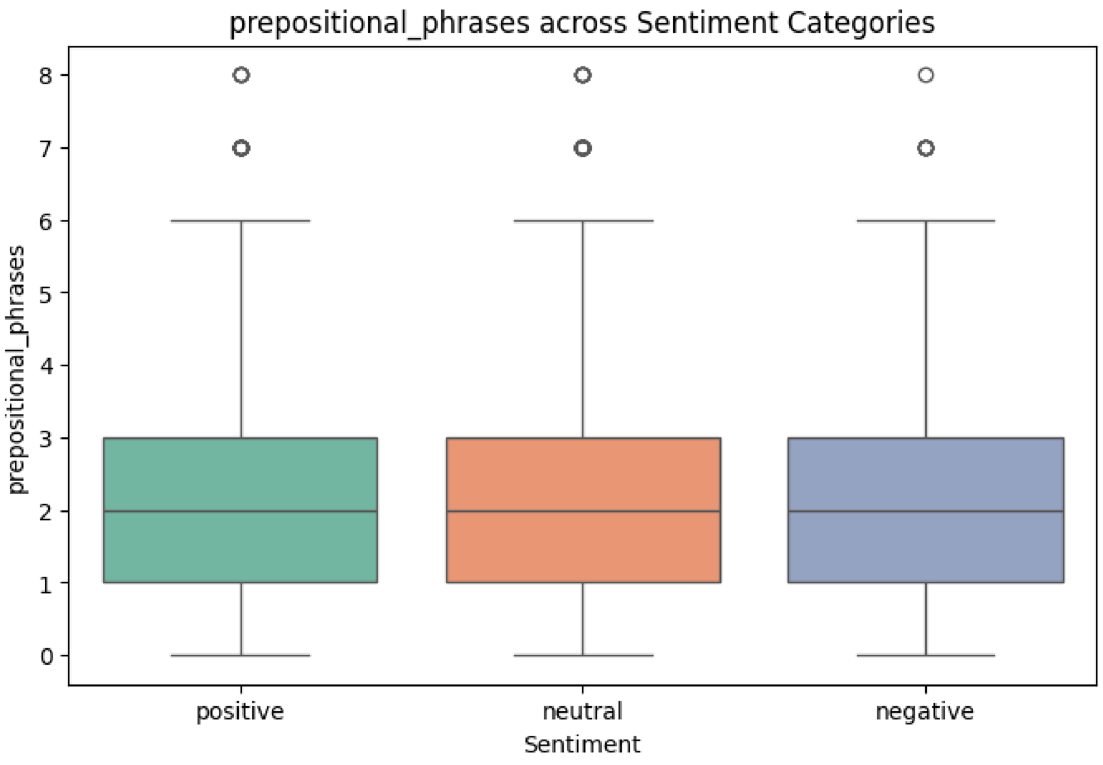
```
<ipython-input-21-cfee21983038>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='sentiment', y=feature, data=df, palette="Set2")
```



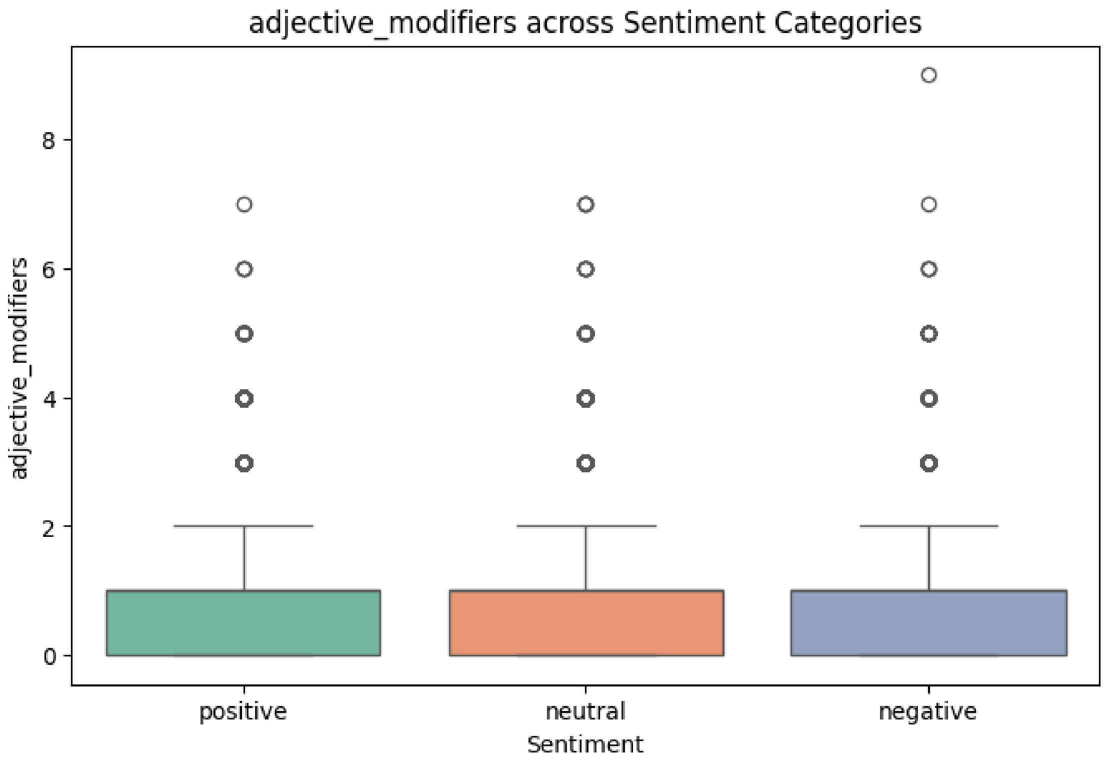
```
<ipython-input-21-cfee21983038>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='sentiment', y=feature, data=df, palette="Set2")
```



```
<ipython-input-21-cfee21983038>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='sentiment', y=feature, data=df, palette="Set2")
```



```
<ipython-input-21-cfee21983038>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='sentiment', y=feature, data=df, palette="Set2")
```



Statistical Testing

To determine whether differences in syntactic features across sentiment categories are statistically significant, I apply the Kruskal-Wallis H test (non-parametric ANOVA).

[22]:
```python
import scipy.stats as stats

print("\nKruskal-Wallis Test Results:")
for feature in feature_columns:
    negative = df[df['sentiment'] == 'negative'][feature]
    neutral = df[df['sentiment'] == 'neutral'][feature]
    positive = df[df['sentiment'] == 'positive'][feature]
    stat, p = stats.kruskal(negative, neutral, positive)
    print(f"{feature}: H = {stat:.3f}, p = {p:.4f}")
```

```
Kruskal-Wallis Test Results:
subordinate_clauses: H = 454.496, p = 0.0000
conjunctions: H = 41.547, p = 0.0000
sentence_length: H = 243.094, p = 0.0000
avg_noun_phrase_length: H = 278.569, p = 0.0000
prepositional_phrases: H = 148.273, p = 0.0000
adjective_modifiers: H = 259.895, p = 0.0000
```

Correlation Analysis

We can also check how each syntactic feature correlates with sentiment polarity.

[23]:
```python
# Map sentiment to numeric polarity
df['sentiment_score'] = df['sentiment'].map({'negative': -1, 'neutral': 0,␣
 ↪'positive': 1})

for feature in feature_columns:
    corr, p = stats.spearmanr(df[feature], df['sentiment_score'])
    print(f"{feature}: Spearman correlation = {corr:.3f}, p = {p:.4f}")
```

```
subordinate_clauses: Spearman correlation = -0.068, p = 0.0000
conjunctions: Spearman correlation = -0.009, p = 0.0650
sentence_length: Spearman correlation = -0.025, p = 0.0000
avg_noun_phrase_length: Spearman correlation = 0.028, p = 0.0000
prepositional_phrases: Spearman correlation = -0.005, p = 0.2755
adjective_modifiers: Spearman correlation = 0.042, p = 0.0000
```

0.4 Discussion

This study explored whether measures of clausal and phrasal complexity can help distinguish between sentiment categories (negative, neutral, and positive) in tweets. By analyzing syntactic features in the **TweetEval Sentiment Analysis dataset**, we found statistically significant differences in structural complexity across sentiments, supporting the idea that syntactic patterns play a role in how sentiment is expressed in social media texts.

0.4.1 Relationship to Previous Research

Prior work in sentiment analysis has primarily emphasized lexical and semantic features (Barbieri et al., 2020; Rosenthal et al., 2019), often overlooking the role of sentence structure. However, studies such as Ghosh et al. (2015) and Zhang et al. (2018) have pointed out that syntactic features can help detect subtle or complex sentiment expressions, especially in informal or sarcastic language. Our findings build on this line of research by providing large-scale empirical evidence that syntactic complexity varies across sentiments in Twitter discourse.

- The higher clausal complexity in **negative tweets** (more subordinate clauses and longer sentences) aligns with prior suggestions that expressing negative opinions may involve more nuanced, qualified, or elaborative language structures (Poria et al., 2020). For instance, negative sentiment might involve explaining dissatisfaction, providing reasons, or contrasting ideas—all of which often require more complex sentence constructions.
- **Positive tweets**, in contrast, were associated with higher use of **adjective modifiers**, supporting the idea that positive sentiment may be expressed more through **descriptive and emotive language**, using adjectives to intensify or elaborate positive qualities.
- **Neutral tweets** tended to feature more **prepositional phrases** and slightly longer noun phrases, possibly reflecting more factual, informational, or context-setting language, which is less emotionally charged and requires structural precision.

0.4.2 Interpretation of Statistical Significance

While the Kruskal-Wallis tests revealed highly significant differences ($\mathbf{p < 0.001}$) across all syntactic features, the Spearman correlation coefficients showed **only weak associations** between individual features and sentiment polarity. This suggests that, although syntactic complexity differs on average between sentiment categories, no single structural feature strongly predicts sentiment on its own.

The large dataset size likely contributed to the detection of small but statistically significant effects. Therefore, while the differences are real and consistent, their practical impact may be modest. This aligns with expectations in natural language processing research, where syntax complements but does not replace the semantic content in understanding sentiment.

0.4.3 Implications for Sentiment Analysis

These findings suggest that integrating syntactic complexity features into sentiment analysis models could provide **additional signals** that enhance classification, particularly in challenging cases such as: - Detecting indirect, sarcastic, or nuanced sentiment. - Improving robustness in noisy, informal social media texts.

However, given the weak correlations, syntactic features are unlikely to serve as **primary predictors**. Instead, they may offer incremental gains when combined with lexical, semantic, and contextual features in more complex, multimodal models.

0.4.4 Limitations

Several limitations should be noted: - Tweets are inherently short and often lack full grammatical structures, which may limit the variability and influence of syntactic complexity. - The use of an automatic dependency parser like **spaCy** may introduce parsing errors, especially on noisy text. - This study focused on a fixed set of syntactic features; additional linguistic features (such as discourse markers, negation, or modal verbs) could also play a role in sentiment expression.

0.4.5 Future Work

Future research could explore: - Combining syntactic complexity with semantic and pragmatic features in joint models. - Applying similar analyses to longer texts, such as reviews or news comments, where syntactic complexity may have a stronger role. - Investigating whether certain combinations of syntactic features (rather than individual features) more reliably distinguish sentiment.

0.5 References

Ghosh, A., Fabbri, A. R., & Muresan, S. (2015). Recognizing Sarcasm in Twitter: A Closer Look. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.

Zhang, L., Wang, S., & Liu, B. (2018). Deep Learning for Sentiment Analysis: A Survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

Rosenthal, S., Farra, N., & Nakov, P. (2019). SemEval-2017 Task 4: Sentiment Analysis in Twitter.