avatar-progress-report

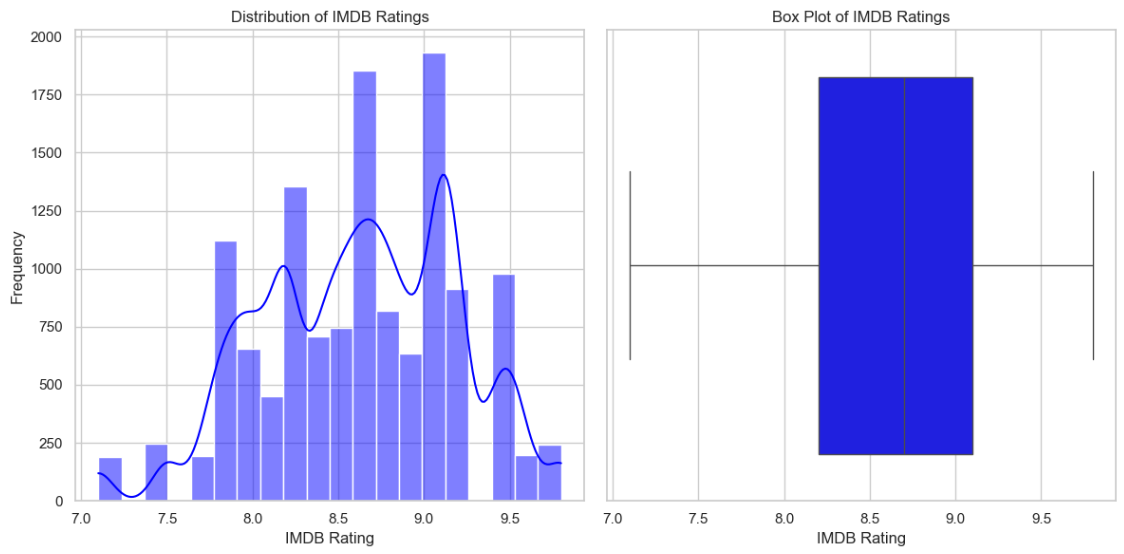
July 18, 2024

[4]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import networkx as nx
from wordcloud import WordCloud
from textblob import TextBlob
import numpy as np

# Load the dataset
file_path = 'avatar.csv'
data = pd.read_csv(file_path, encoding='ISO-8859-1')

# Set the plot style
sns.set(style="whitegrid")

# 1. Distribution of IMDB Ratings
plt.figure(figsize=(12, 6))

# Histogram
plt.subplot(1, 2, 1)
sns.histplot(data['imdb_rating'], bins=20, kde=True, color='blue')
plt.title('Distribution of IMDB Ratings')
plt.xlabel('IMDB Rating')
plt.ylabel('Frequency')

# Box plot
plt.subplot(1, 2, 2)
sns.boxplot(x=data['imdb_rating'], color='blue')
plt.title('Box Plot of IMDB Ratings')
plt.xlabel('IMDB Rating')

plt.tight_layout()
plt.show()

# Set the plot style
sns.set(style="whitegrid")
```

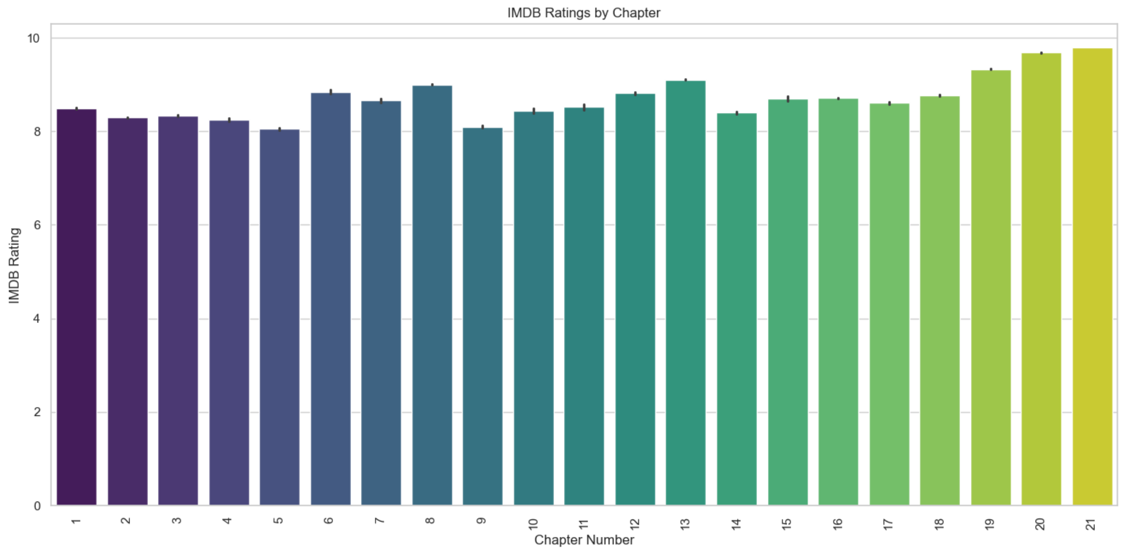
```
# 1. IMDB Ratings by Chapter
plt.figure(figsize=(14, 7))
sns.barplot(x='chapter_num', y='imdb_rating', data=data, palette='viridis')
plt.title('IMDB Ratings by Chapter')
plt.xlabel('Chapter Number')
plt.ylabel('IMDB Rating')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

# 2. IMDB Ratings by Book
plt.figure(figsize=(14, 7))
sns.histplot(data, x='imdb_rating', hue='book', multiple='stack',␣
 ↪palette='viridis', kde=True)
plt.title('IMDB Ratings by Book')
plt.xlabel('IMDB Rating')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()

# 3. IMDB Ratings by Director
# Group by director and calculate mean and count of IMDB ratings
director_ratings = data.groupby('director')['imdb_rating'].agg(['mean',␣
 ↪'count']).reset_index()
director_ratings = director_ratings.sort_values(by='mean', ascending=False)

plt.figure(figsize=(14, 7))
sns.barplot(x='mean', y='director', data=director_ratings, palette='viridis',␣
 ↪orient='h')
plt.title('Average IMDB Ratings by Director')
plt.xlabel('Average IMDB Rating')
plt.ylabel('Director')
plt.tight_layout()
plt.show()

plt.figure(figsize=(14, 7))
sns.barplot(x='count', y='director', data=director_ratings, palette='viridis',␣
 ↪orient='h')
plt.title('Number of Episodes Directed by Each Director')
plt.xlabel('Number of Episodes')
plt.ylabel('Director')
plt.tight_layout()
plt.show()

# 2. Character-based Word Clouds and Sentiment Analysis

# Generate word clouds and sentiment analysis for each character
```

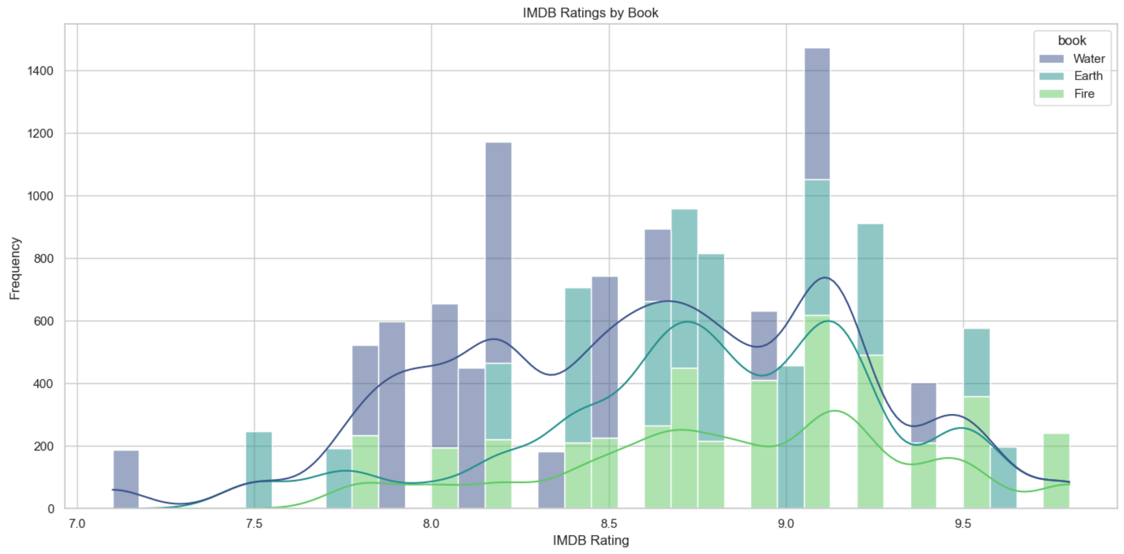
```
characters = ['Aang', 'Katara', 'Zuko', 'Sokka', 'Toph', 'Iroh', 'Azula']

for character in characters:
    character_data = data[data['character'] == character]

    if not character_data['character_words'].dropna().empty:
        # Generate word cloud
        character_text = ' '.join(character_data['character_words'].dropna().
 ↪astype(str).tolist())
        wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate(character_text)

        # Plotting the word cloud
        plt.figure(figsize=(12, 6))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis('off')
        plt.title(f'Word Cloud for {character}')
        plt.show()

        # Sentiment analysis using TextBlob
        character_data['sentiment'] = character_data['character_words'].
 ↪dropna().apply(lambda x: TextBlob(x).sentiment.polarity)

        # Plotting the sentiment analysis
        plt.figure(figsize=(12, 6))
        sns.histplot(character_data['sentiment'], bins=30, kde=True,␣
 ↪color='purple')
        plt.title(f'Sentiment Analysis for {character}')
        plt.xlabel('Sentiment Score')
        plt.ylabel('Frequency')
        plt.show()

# 3. Simplified Chapter Analysis

# Average IMDB rating per chapter
chapter_ratings = data.groupby('chapter')['imdb_rating'].mean().reset_index()

# Number of lines per chapter
chapter_lines = data['chapter'].value_counts().reset_index()
chapter_lines.columns = ['chapter', 'line_count']

# Simplified chapter plot
plt.figure(figsize=(12, 6))

# Average IMDB rating per chapter
plt.subplot(1, 2, 1)
```

```python
sns.barplot(x=chapter_ratings['chapter'], y=chapter_ratings['imdb_rating'],␣
 ↪palette='coolwarm')
plt.title('Average IMDB Rating per Chapter')
plt.xlabel('Chapter')
plt.ylabel('Average IMDB Rating')
plt.xticks(rotation=90)

# Number of lines per chapter
plt.subplot(1, 2, 2)
sns.barplot(x=chapter_lines['chapter'], y=chapter_lines['line_count'],␣
 ↪palette='coolwarm')
plt.title('Number of Lines per Chapter')
plt.xlabel('Chapter')
plt.ylabel('Number of Lines')
plt.xticks(rotation=90)

plt.tight_layout()
plt.show()

# List of important characters
important_characters = ['Aang', 'Katara', 'Zuko', 'Sokka', 'Toph', 'Iroh',␣
 ↪'Azula']

# Filter data to only include interactions between important characters
filtered_data = data[data['character'].isin(important_characters)]

# Create a co-occurrence matrix
co_occurrence = pd.crosstab(filtered_data['chapter'],␣
 ↪filtered_data['character'])

# Compute the co-occurrence matrix
interaction_matrix = co_occurrence.T.dot(co_occurrence)
interaction_matrix = interaction_matrix.where(~np.eye(interaction_matrix.
 ↪shape[0], dtype=bool))

# Build the interaction graph
G = nx.Graph()

# Add nodes
for character in important_characters:
    G.add_node(character)

# Add edges
for i, character1 in enumerate(important_characters):
    for j, character2 in enumerate(important_characters):
        if i < j and interaction_matrix.loc[character1, character2] > 0:
            G.add_edge(character1, character2, weight=interaction_matrix.
↪loc[character1, character2])

# Visualize the graph
plt.figure(figsize=(12, 12))
pos = nx.spring_layout(G, k=0.5)
nx.draw(G, pos, with_labels=True, node_size=3000, node_color='skyblue',␣
↪edge_color='grey', font_size=12, font_weight='bold')
nx.draw_networkx_edge_labels(G, pos, edge_labels={(u, v): f'{d["weight"]}' for␣
↪u, v, d in G.edges(data=True)}, font_color='red')
plt.title('Character Interaction Network Graph')
plt.show()
```

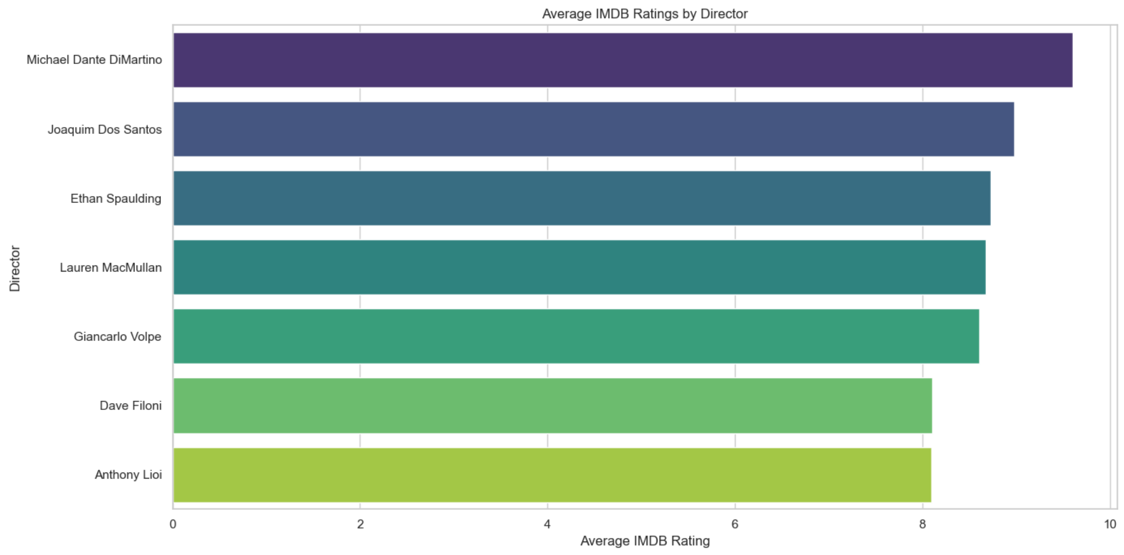


```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:40: FutureWarning:

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.barplot(x='chapter_num', y='imdb_rating', data=data, palette='viridis')
```

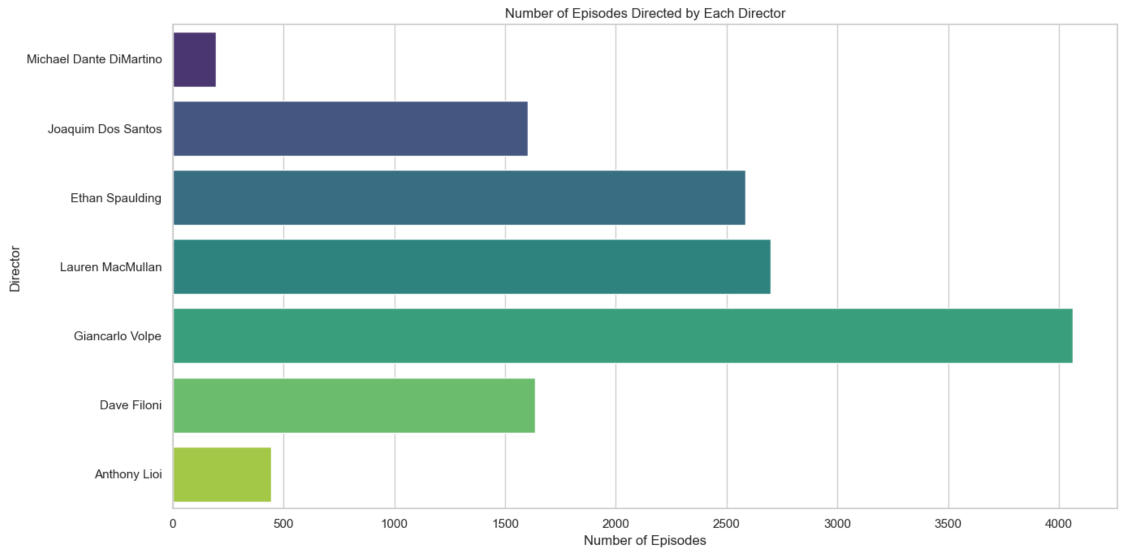


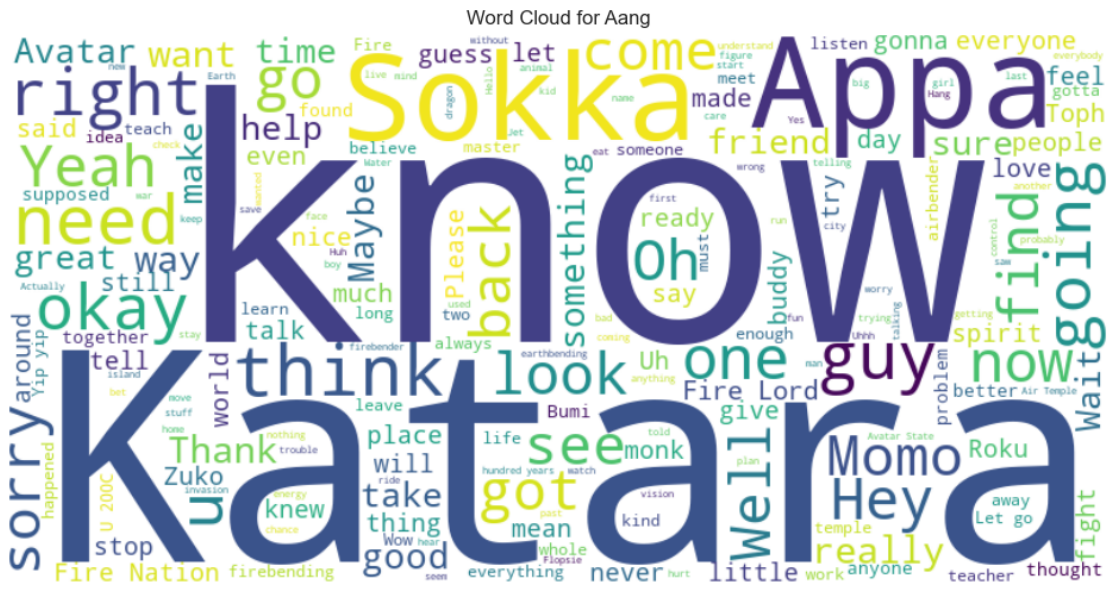


```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:63: FutureWarning:

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

  sns.barplot(x='mean', y='director', data=director_ratings, palette='viridis',
orient='h')
```

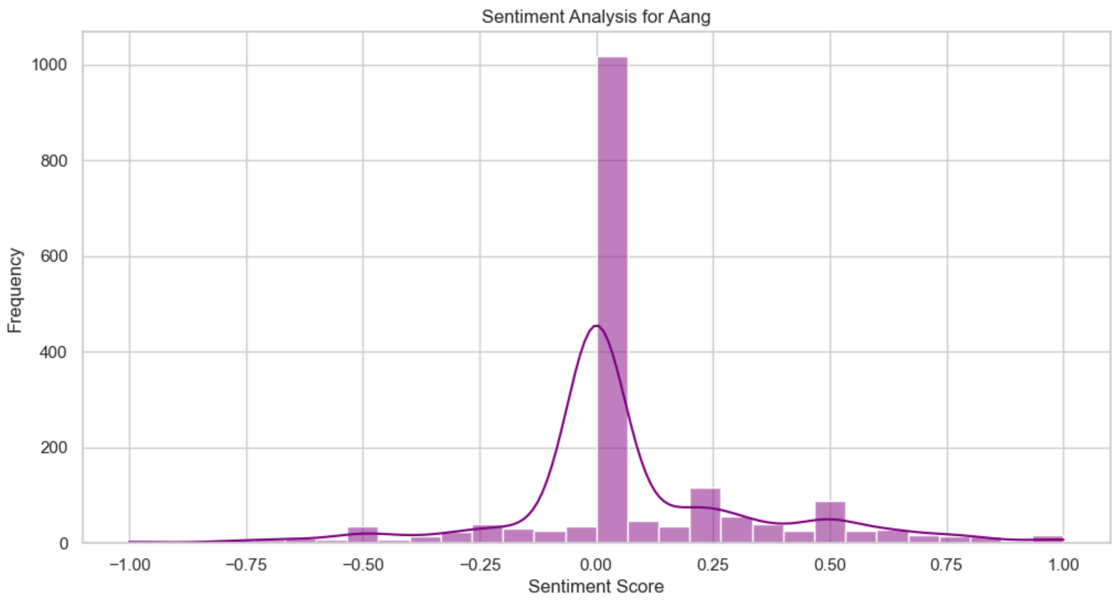


```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:71: FutureWarning:

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

  sns.barplot(x='count', y='director', data=director_ratings, palette='viridis',
orient='h')
```

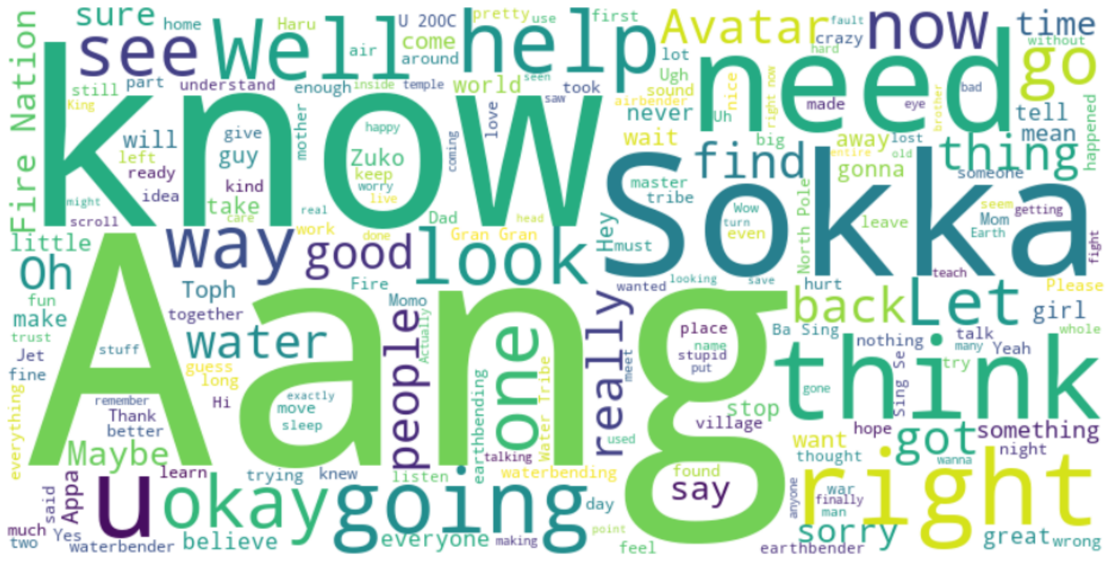




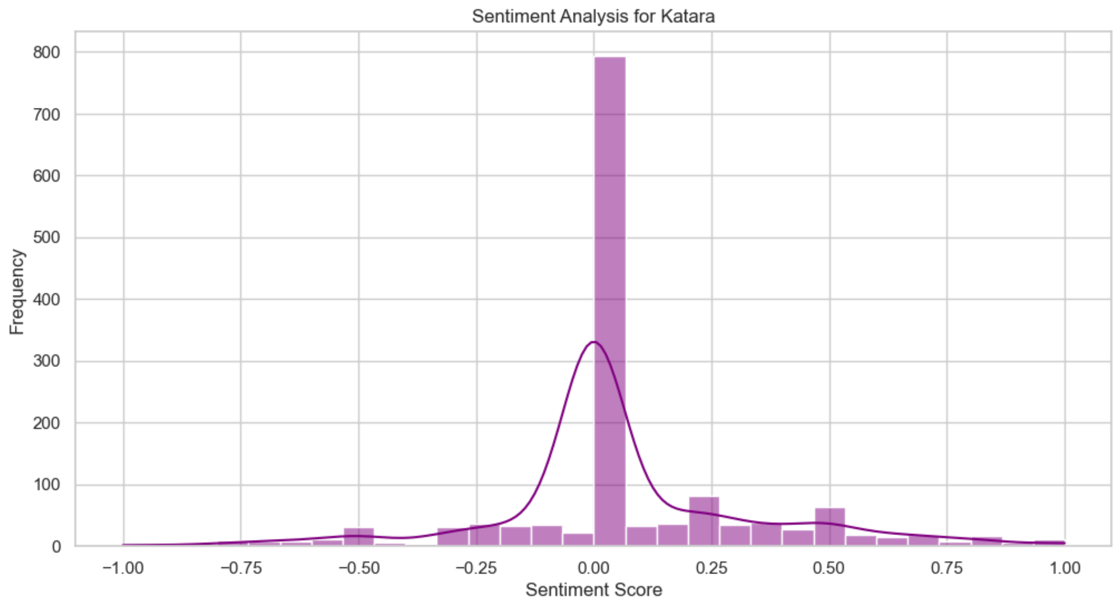
```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:99:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  character_data['sentiment'] =
character_data['character_words'].dropna().apply(lambda x:
TextBlob(x).sentiment.polarity)
```

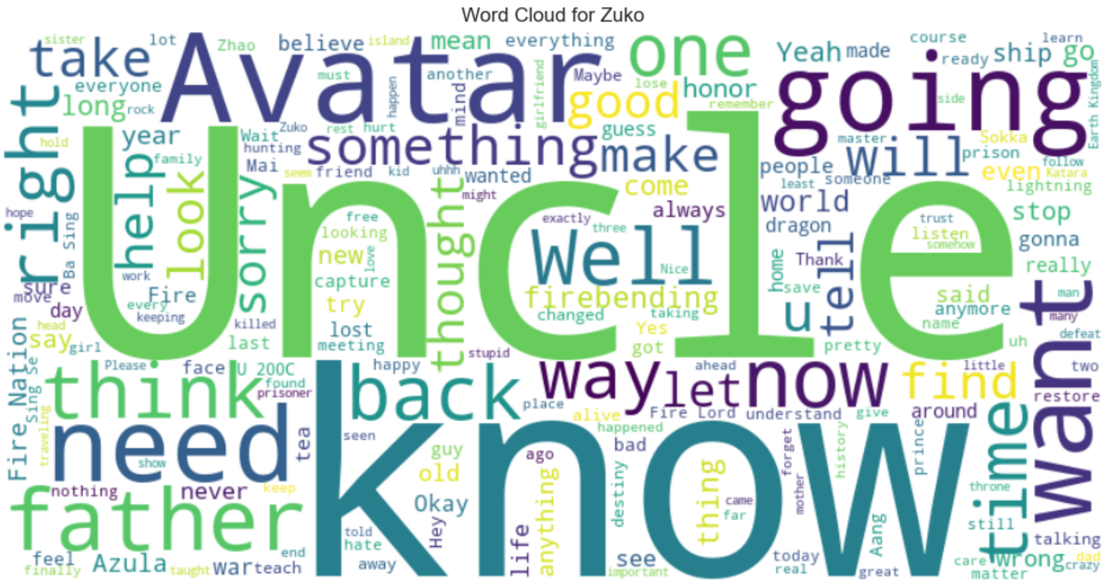


Word Cloud for Katara

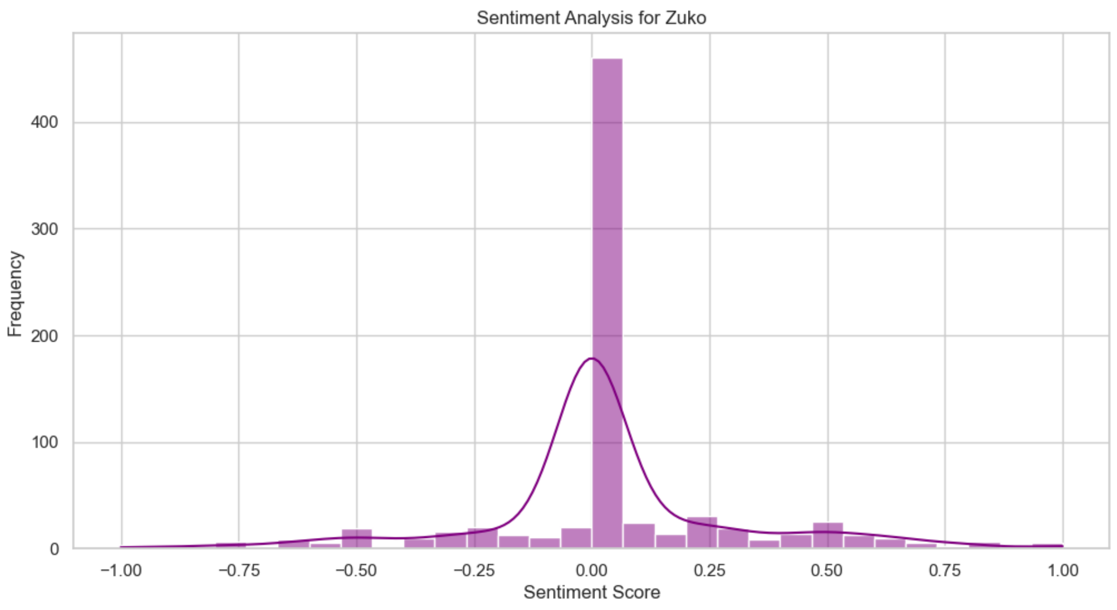


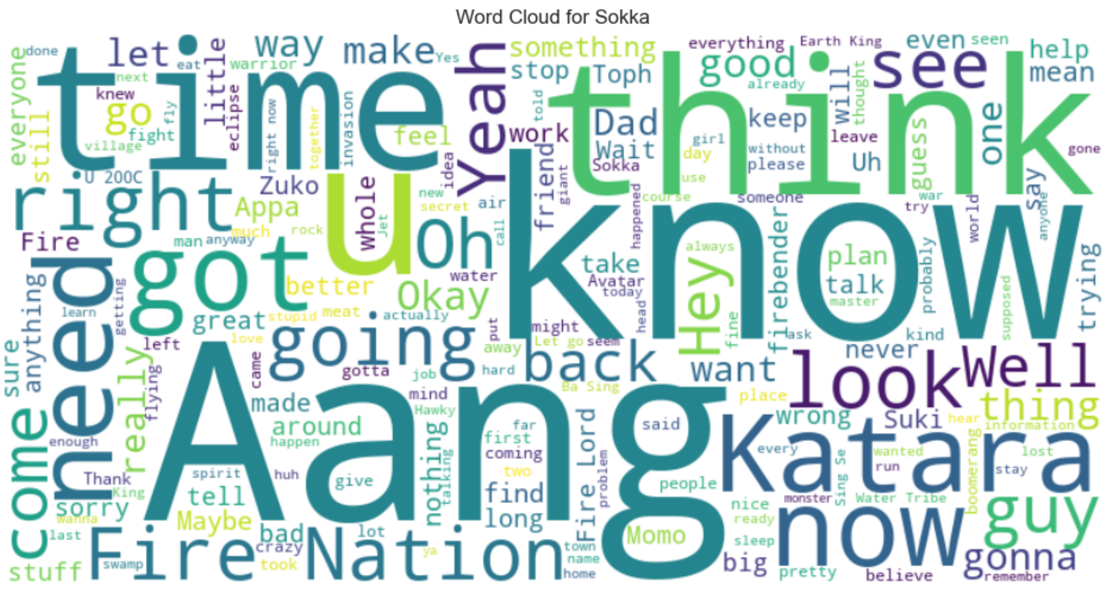
```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:99:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  character_data['sentiment'] =
character_data['character_words'].dropna().apply(lambda x:
TextBlob(x).sentiment.polarity)
```





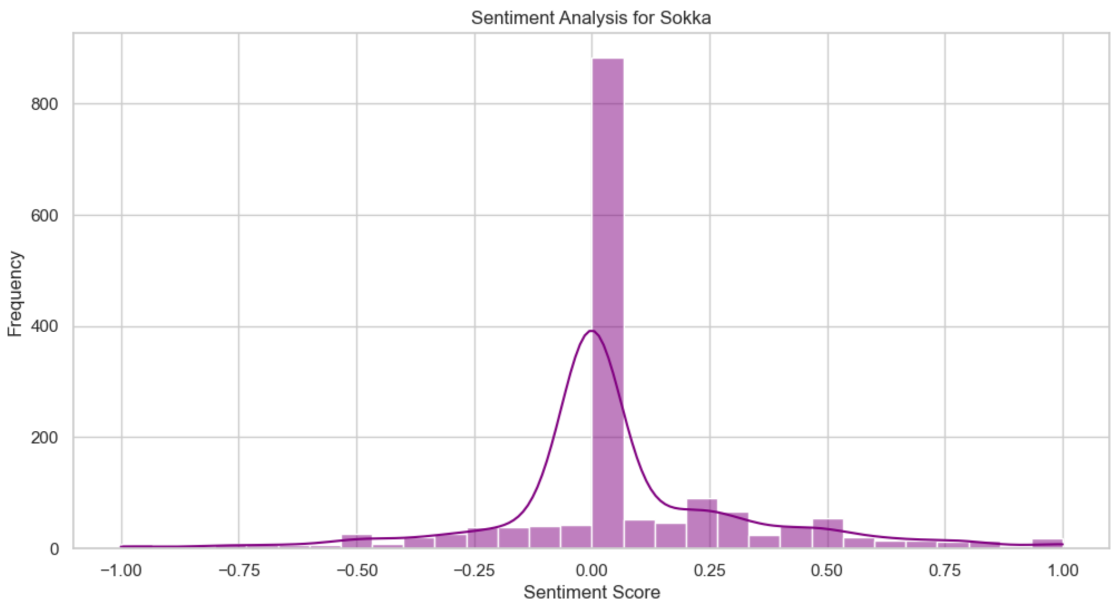
```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:99:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  character_data['sentiment'] =
character_data['character_words'].dropna().apply(lambda x:
TextBlob(x).sentiment.polarity)
```

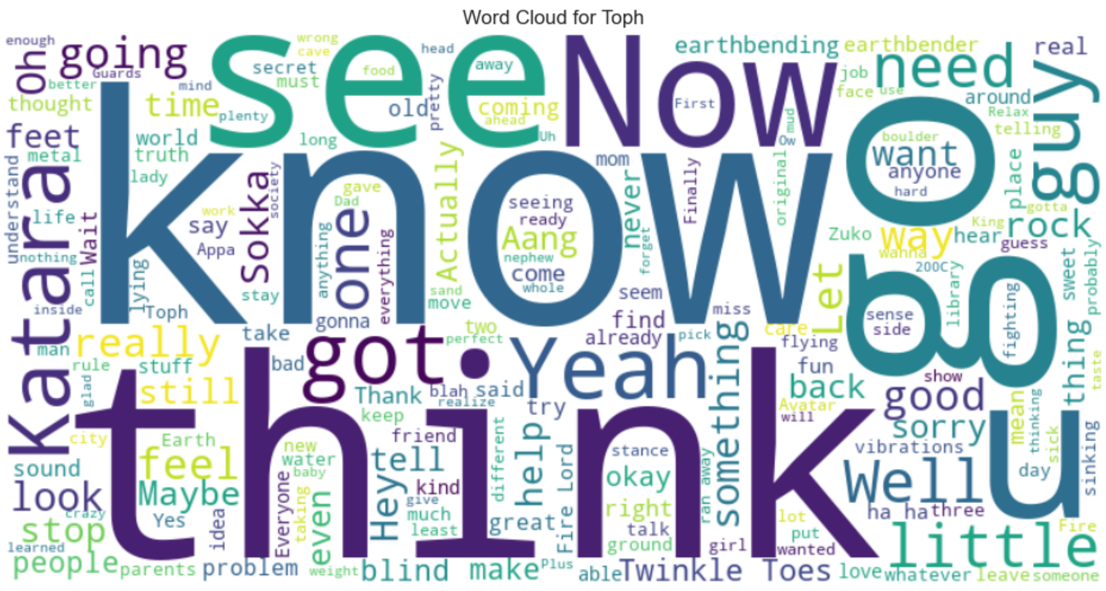




```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:99:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

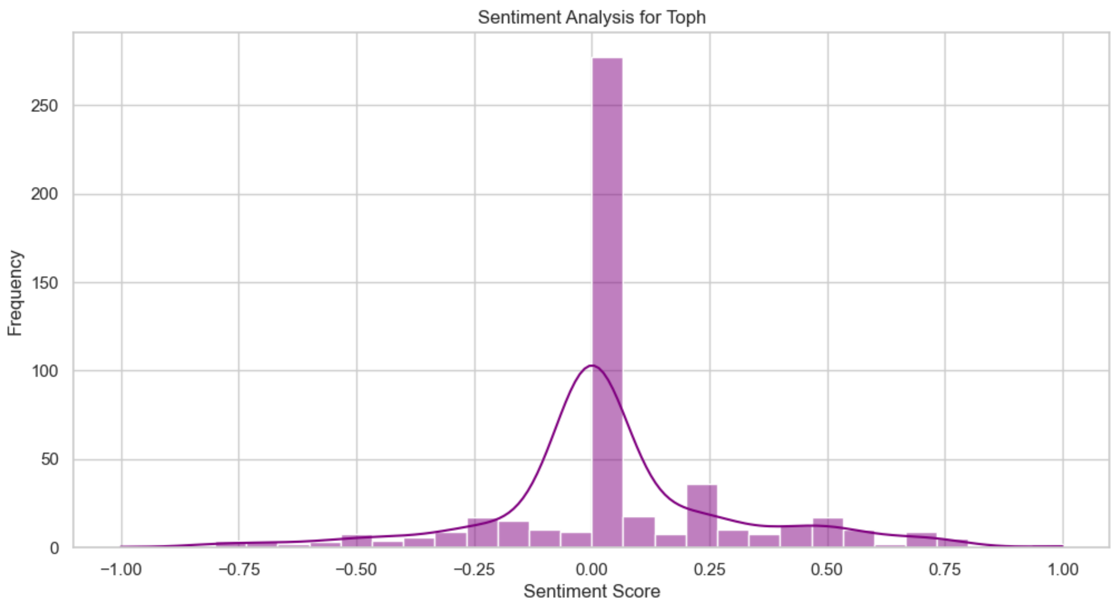
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  character_data['sentiment'] =
character_data['character_words'].dropna().apply(lambda x:
TextBlob(x).sentiment.polarity)
```

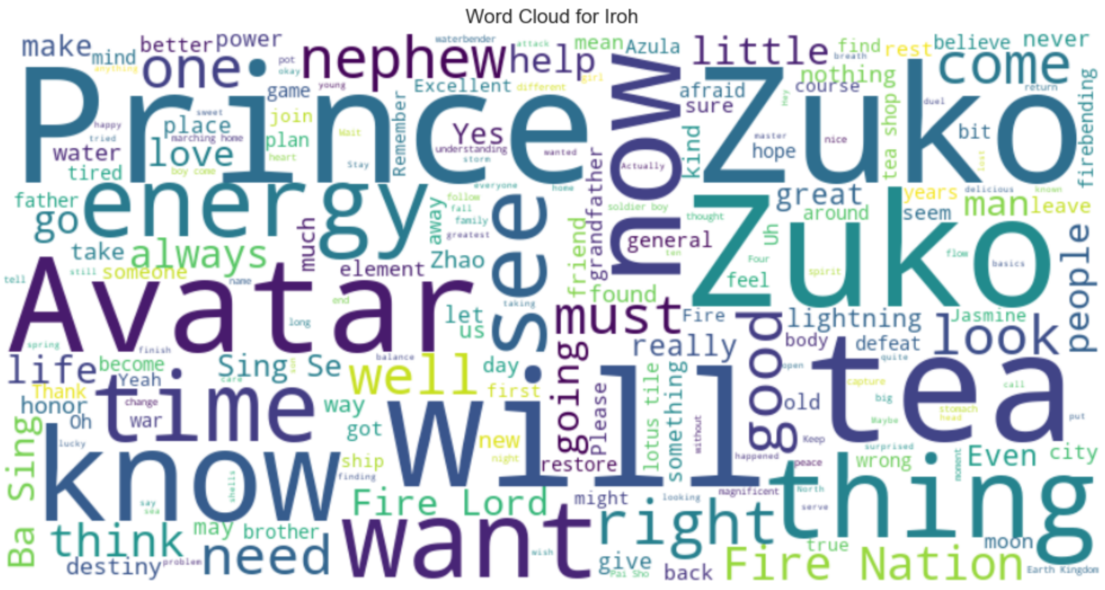




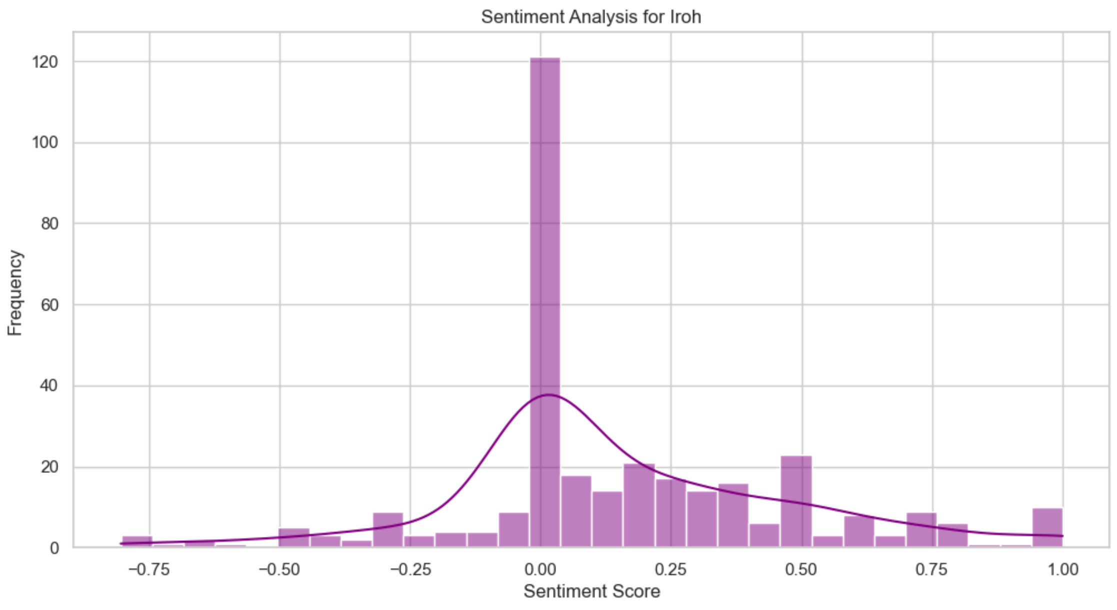
```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:99:
SettingWithCopyWarning:
```

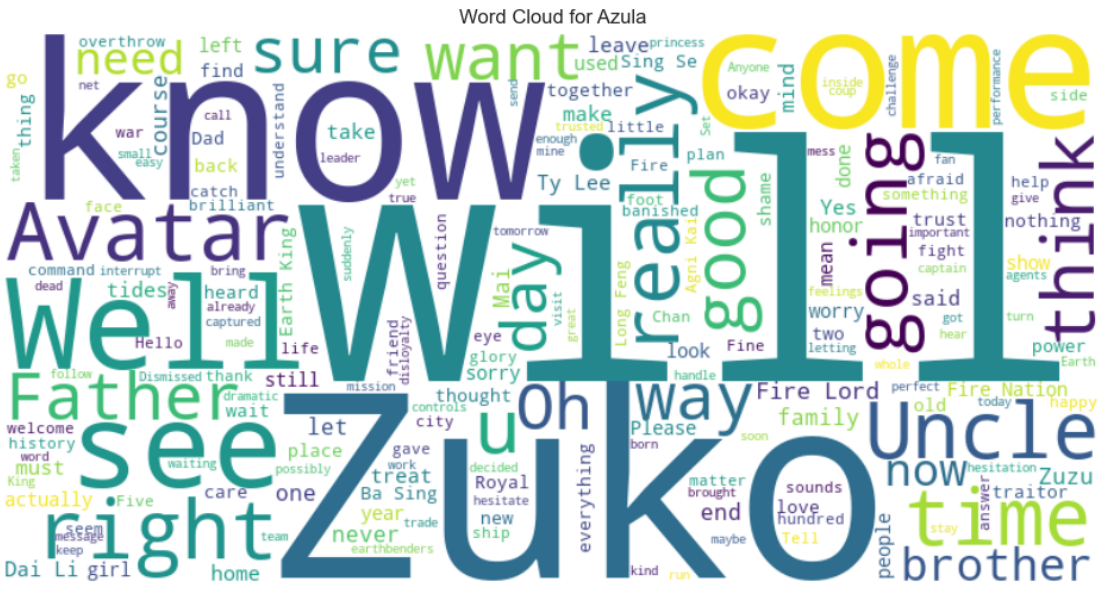
```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  character_data['sentiment'] =
character_data['character_words'].dropna().apply(lambda x:
TextBlob(x).sentiment.polarity)
```



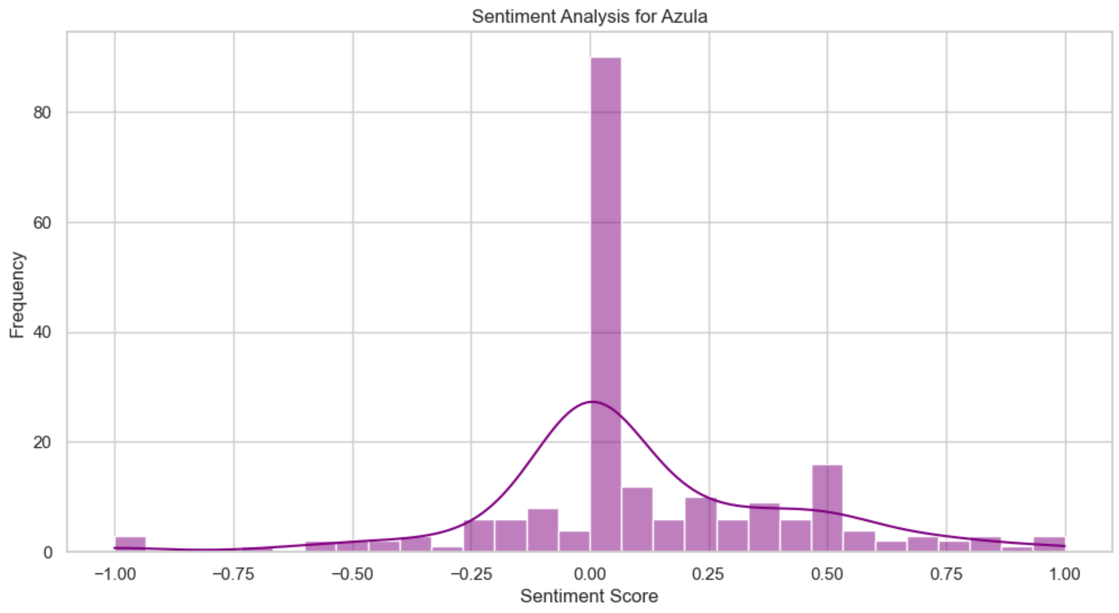


```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:99:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  character_data['sentiment'] =
character_data['character_words'].dropna().apply(lambda x:
TextBlob(x).sentiment.polarity)
```



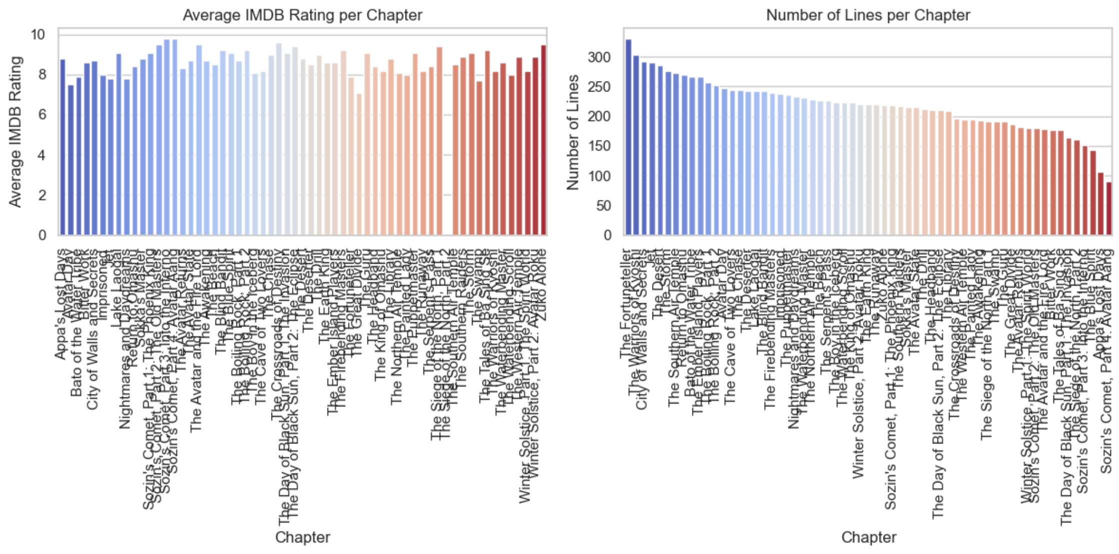


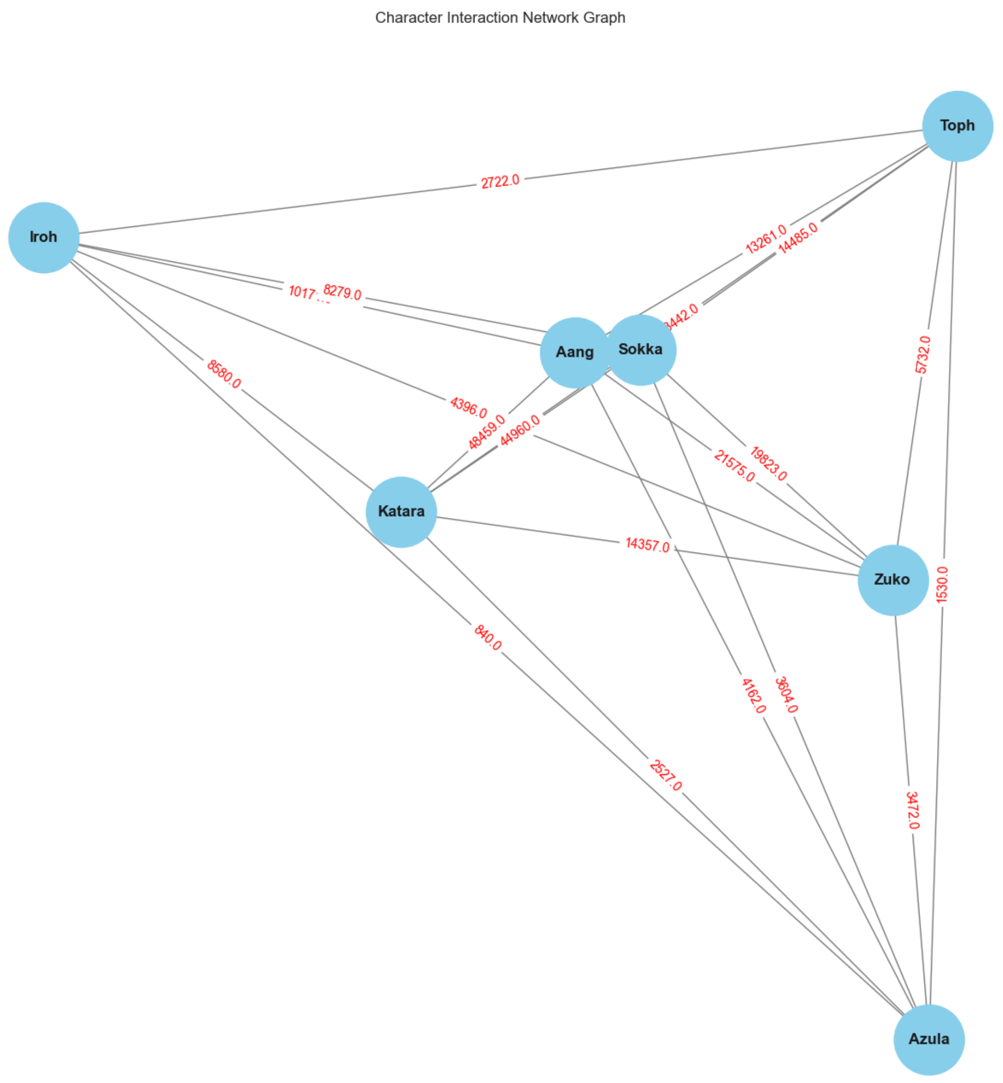
```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:99:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  character_data['sentiment'] =
character_data['character_words'].dropna().apply(lambda x:
TextBlob(x).sentiment.polarity)
```



```
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:123:
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.barplot(x=chapter_ratings['chapter'], y=chapter_ratings['imdb_rating'],
palette='coolwarm')
C:\Users\ozgeo\AppData\Local\Temp\ipykernel_11396\92806494.py:131:
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.barplot(x=chapter_lines['chapter'], y=chapter_lines['line_count'],
palette='coolwarm')
```




Character Interaction Network Graph
Toph
Iroh
Aang
Sokka
Katara
Zuko
Azula
2722.0
13261.0
14485.0
442.0
1017
8279.0
8580.0
4396.0
48459.0
44960.0
19823.0
21575.0
5732.0
14357.0
1530.0
840.0
4162.0
3604.0
2527.0
3472.0