Technical Report: Final Project DS 5110: Introduction to Data Management and Processing

Team Members:
Khoury College of Computer Sciences
Fei Li
li.fei4@northeastern.edu
Chenzi Qian
qian.chenz@northeastern.edu

November 18, 2024

Contents

1 Introduction

In this project, we analyze the popular TV show Friends to uncover statistical insights using a variety of data processing libraries. Specifically, we focus on examining how guest stars influence the show's ratings, including audience votes and star evaluations.

This analysis encompasses data collection, processing, and interpretation, demonstrating how meaningful insights can be derived from structured datasets. By exploring the relationship between guest appearances and audience reception, the project aims to provide a deeper understanding of factors contributing to the show's enduring popularity.

2 Literature Review

Previous studies on TV shows have primarily focused on aspects such as episode views, show duration, and the analysis of actor lines spoken. Research has shown how factors like episode length and dialogue frequency influence audience engagement and ratings. Additionally, studies have often explored the impact of primary cast members' screen time and performance on the reception of a show. Some studies also analyze specific episode traits like plot development and character involvement, which are often linked to audience ratings.

However, there is a noticeable gap in research concerning the direct impact of guest stars on TV show ratings and audience engagement. While star power and celebrity influence have been explored in other media contexts, the specific relationship between guest star appearances and the show's ratings—especially audience votes and star ratings—remains underexplored.

In this project, we seek to address this gap by examining how guest star appearances influence show success, with a focus on audience votes and star ratings. Additionally, we aim to compare ratings from IMDb with votes cast by viewers to analyze potential differences in audience perception between professional critics and the general public. This will provide a more comprehensive view of how guest stars and other factors influence the reception of the show.

3 Methodology

3.1 Data Collection

The data for this project was collected from several sources to provide a comprehensive analysis of the episodes and celebrity appearances in Friends.

Episode Information: For the overall episode details, we sourced a dataset from Kaggle, which includes essential information such as episode titles, episode numbers, and season numbers. This data was provided in CSV format. We used Google Colab as our primary tool for code editing and analysis. Using Python's pandas library, we read the CSV file and loaded the data into a DataFrame for further processing and analysis.

Celebrity Guest Information: For celebrity appearances, we initially sourced a Google Sheet file containing celebrity data, including names and guest appearances in the series, from an IMDb report. However, the data was incomplete, as some entries only included the names of the guest stars without specific season or episode information. To address this gap, we personally verified the information by reviewing each celebrity's episodes to

ensure data integrity. We then supplemented the dataset by manually adding the missing details. This additional data was compiled into a Google Sheet, which was later exported as a CSV file for integration into our analysis.

3.2 Data Preprocessing

The data used in this project required several steps of cleaning and preprocessing to ensure its integrity and consistency before analysis. The following steps were taken:

Loading the Data: After collecting the data from various sources (Kaggle for episode details and Google Sheets for celebrity guest information), the first step was to load the data into Python using the pandas library. This was done by reading the CSV files into DataFrames for easy manipulation.

Creating Needed DataFrames: We created several different DataFrames based on our specific needs for analysis. These included separate DataFrames for episode information, guest star appearances, and any other relevant datasets.

Merging the Datasets: After cleaning, the episode and celebrity guest datasets were merged based on common identifiers, such as episode numbers and season numbers. This step ensured that we had a unified dataset containing both episode details and the corresponding celebrity guest information.

Creating New Fields: In some cases, we created new features from the existing data to enrich the dataset. For example, we created an "episode index" field in the overall DataFrame to better organize and manipulate the data.

Data Cleaning: After merging, we found that some fields contained NaN values. To handle this, we inserted appropriate default values where needed to ensure no NaN values remained. For example, we set the "Audience Cheering" field to "n" by default if it was NaN, and we set guest star names to "no_exist" if they were missing.

To confirm that all NaN values were properly handled, we used the isnull().sum() method to check for any remaining missing data.

3.3 Analysis Techniques

The analysis was divided into two main components. First, generalized analysis focused on trends in audience engagement using images to visualize votes and star ratings over time. Bar charts highlighted the top episodes by votes and star ratings, while Venn diagrams illustrated the overlap between highly rated and highly voted episodes. For sentiment analysis, we are considering using pre-trained models to compute sentiment polarity scores for each dialogue, which were aggregated by character and season to identify emotional trends. Additionally, topic modeling was conducted using Latent Dirichlet Allocation (LDA) to uncover recurring themes in dialogues, which were then analyzed to draw insights into character-specific and season-wide patterns.

4 Results

The results so far show some interesting patterns in the data. From the generalized analysis, we found that audience votes and star ratings stayed fairly steady across seasons, but there were noticeable spikes for certain popular episodes like "The One with the Prom Video" from Season 2 and "The Last One" from Season 10. Episodes with guest stars tended to get higher audience engagement, especially when the audience was cheering.

For example, episodes featuring Brad Pitt stood out. On average, star ratings seemed to peak during the middle seasons, suggesting these were the most loved by viewers.

In the sentiment analysis, early results show that Rachel and Monica generally had positive emotions in their dialogues, which matches their supportive roles in the group. Chandler's lines were often neutral or slightly negative, fitting with his sarcastic sense of humor. Emotional lows in the dialogue aligned with major story events like Ross and Rachel's breakup. Topic modeling revealed recurring themes like love, friendship, and humor, with Joey's lines often focused on relationships and acting, while Phoebe's reflected her creative and quirky personality. These results are promising but need more work to draw stronger conclusions.

5 Discussion

These early findings show that combining different types of analysis can give us valuable insights into the show Friends. The patterns in audience votes and ratings highlight how well the show balanced humor, emotional depth, and nostalgia. The sentiment analysis shows that the emotional tone of the characters' dialogues was an important part of their appeal to the audience. Themes found in the dialogues also reflect the unique personalities of the characters and the universal topics the show explored, like relationships and friendships.

However, this project is not finished, and there are still areas that need improvement. For example, the sentiment analysis relies on a pre-trained model that may not capture the show's sarcasm or humor very well. The guest star analysis could also be expanded by looking at additional data, like audience demographics or social media reactions, to better understand why certain episodes were more popular. We are continuing to refine the analysis to make the results more accurate and meaningful.

6 Conclusion

This project has made progress in analyzing Friends through data, but it is still a work in progress. So far, we've found interesting patterns in audience engagement, character emotions, and recurring themes. The results suggest that the show's success comes from a mix of relatable emotions, strong characters, and entertaining guest appearances.

At the same time, there is more to do to improve the analysis. Better tools for understanding sarcasm and humor in the dialogue could help refine the sentiment analysis. Adding more data, like social media reactions, could give us a fuller picture of how the audience responded to different episodes. As we continue to work on this project, we hope to uncover even deeper insights into what made Friends such a beloved show.

7 References

References

Hutto, C. J., & Gilbert, E. (2014). "VADER: A Sentiment Analysis Model." Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). "Latent Dirichlet Allocation."

A Appendix A: Code

Include any relevant code used in the project. For example:

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

df1 = pd.read_csv("/content/friends_episodes_v3.csv")
df2 = pd.read_csv("/content/guest_stars_info.csv")

# concat the df together

# DF_ALL = pd.concat([df1,df2])
# print(DF_ALL.head(2))

# Join the data frames to one

all_episodes_info = pd.merge(df1, df2, on=['Season','Episode_Number'], how='outer')
all_episodes_info['Episode_Index'] = range(1, len(all_episodes_info) + 1)

episodes_celebrities_df = all_episodes_info[all_episodes_info['Guest_Star_Name'].notna()]
episodes_celebrities_cheering_df = episodes_celebrities_df[episodes_celebrities_df['Audience_Cheering'] == 'y']

# Merge episodes_only_with_celebrities with all_episodes_info to add the Episode_Index
print("before merging........")
print(len(episodes_celebrities_df))
#
print(episodes_celebrities_df.head(7))

#Tasks
# for data cleaning, check null value, also , if there is NaN, replace with reasonable values instead
# Check for missing values (NaN) in both DataFrames
print("Missing values in all_episodes_info:")
print(all_episodes_info.isnull().sum())


# For other columns, you can decide based on your domain knowledge if you want to fill them with a specific value
all_episodes_info['Guest_Star_Name'] = all_episodes_info['Guest_Star_Name'].fillna('Non_Exist')
all_episodes_info['Audience_Cheering'] = all_episodes_info['Audience_Cheering'].fillna('n')


# If you want to drop rows with missing values entirely (use cautiously)
# episodes_only_with_celebrities.dropna(inplace=True)

# Verify that missing values were filled
print("Missing values after filling in episodes_only_with_celebrities:")
```

```
print(episodes_celebrities_df.isnull().sum())

#Tasks
# for data cleaning, check null value, also , if there is NaN, replace with reasonable values instead
# Check for missing values (NaN) in both DataFrames
print("Missing values in all_episodes_info:")
print(all_episodes_info.isnull().sum())


# For other columns, you can decide based on your domain knowledge if you want to fill them with a specific value
all_episodes_info['Guest_Star_Name'] = all_episodes_info['Guest_Star_Name'].fillna('Non_Exist')
all_episodes_info['Audience_Cheering'] = all_episodes_info['Audience_Cheering'].fillna('n')


# Verify that missing values were filled
print("Missing values after filling in episodes_only_with_celebrities:")
print(episodes_celebrities_df.isnull().sum())

# Plot a scatter plot to show  Votes trend

plt.figure(figsize=(10, 6))

# Scatter plot for Votes
sns.scatterplot(data=all_episodes_info, x='Episode_Index', y='Votes', color='blue')

# Add titles and labels
plt.title("Votes trend", fontsize=16)
plt.xlabel("Episode_Index", fontsize=14)
plt.ylabel("Votes", fontsize=14)

# Show the plot
plt.show()

# Plot a scatter plot to show  Votes trend

plt.figure(figsize=(10, 6))

# Scatter plot for Votes
sns.scatterplot(data=all_episodes_info, x='Episode_Index', y='Votes', color='blue')

# Add titles and labels
plt.title("Votes trend", fontsize=16)
plt.xlabel("Episode_Index", fontsize=14)
plt.ylabel("Votes", fontsize=14)

# Show the plot
plt.show()

# based on votes , find the top 8 episodes, display the episode number and season number , and episode title , display using a
# graph
```

```
94
95 #Sort the episodes based on the number of votes in descending order
96 top_8_episodes_votes = all_episodes_info.sort_values(by='Votes',
       ascending=False).head(8)
97
98 top_8_episodes_votes_info = top_8_episodes_votes[['Episode_Index', '
       Season', 'Episode_Number', 'Episode_Title', 'Votes']]
99
100 #Create the plot
101 plt.figure(figsize=(10, 6))
102
103 # Bar plot showing the top 8 episodes by number of votes
104 sns.barplot(x='Votes', y='Episode_Title', data=
       top_8_episodes_votes_info)
105
106 # Add titles and labels
107 plt.title("Top 8 Episodes Based on Votes", fontsize=16)
108 plt.xlabel("Number of Votes", fontsize=14)
109 plt.ylabel("Episode Title", fontsize=14)
110
111 # Step 4: Add episode number and season number inside the bars
112 for idx, (i, row) in enumerate(top_8_episodes_votes_info.iterrows()):
113     episode_label = f"Ep {row['Episode_Number']} - Season {row['Season
       ']}"
114     # Position the label in the middle of the bar
115     x_pos = row['Votes'] / 2
116     # Place the label inside the bar
117     plt.text(x_pos, idx, episode_label,
118              color='white',
119              ha='center',
120              va='center',
121              fontsize=10, fontweight='bold')
122
123 # Adjust the plot limits to accommodate the labels
124 plt.xlim(0, top_8_episodes_votes_info['Votes'].max() * 1.2)  # Extend x
       -axis by 20% to fit labels
125
126 # Manually adjust margins to ensure tight layout works
127 # plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.2)  #
       Adjust margins
128
129 # Show the plot
130 plt.show()
131
132 # top 8 episodes by Stars
133 # based on stars , find the top 8 episodes, display the episode number
       and season number , and episode title , display using a
134 # graph
135
136 # Step 1: Sort the episodes based on the star ratings in descending
       order
137 top_8_episodes_Stars = all_episodes_info.sort_values(by='Stars',
       ascending=False).head(8)
138
139 # Step 2: Create a new DataFrame with the relevant columns (
       Episode_Index, Season, Episode_Number, Title, Stars)
140 top_8_episodes_info_Stars = top_8_episodes_Stars[['Episode_Index', '
       Season', 'Episode_Number', 'Episode_Title', 'Stars']]
```

```
141
142 # Step 3: Create the plot
143 plt.figure(figsize=(12, 6))  # Increase the figure size for better
    visualization
144
145 # Apply a color palette based on the 'Stars' rating
146 # We will use a color palette that assigns colors from blue (lower
    stars) to red (higher stars)
147 sns.barplot(x='Stars', y='Episode_Title',hue="Stars", data=
    top_8_episodes_info_Stars, palette='coolwarm')
148
149 # Add titles and labels
150 plt.title("Top 8 Episodes Based on Stars", fontsize=16)
151 plt.xlabel("Star Ratings", fontsize=14)
152 plt.ylabel("Episode Title", fontsize=14)
153
154 # Add episode number and season number inside the bars
155 for idx, (i, row) in enumerate(top_8_episodes_info_Stars.iterrows()):
156     episode_label = f"Ep {row['Episode_Number']} - Season {row['Season
    ']}"
157
158     # Position the label in the middle of the bar
159     x_pos = row['Stars'] / 2
160
161     # Place the label inside the bar
162     plt.text(x_pos, idx, episode_label,
163              color='white',
164              ha='center',
165              va='center',
166              fontsize=12,
167              fontweight='bold')
168
169 # Show the plot
170 plt.show()
171
172 # Plot a scatter plot to show  Stars trend
173
174 plt.figure(figsize=(10, 6))
175
176 # Scatter plot for Votes
177 sns.scatterplot(data=all_episodes_info, x='Episode_Index', y='Stars',
    color='red')
178
179 # Add titles and labels
180 plt.title("Stars Trend", fontsize=16)
181 plt.xlabel("Episode_Index", fontsize=14)
182 plt.ylabel("Stars", fontsize=14)
183
184 # Show the plot
185 plt.show()
186
187 # Calculate the average Stars by Season
188 season_trend = all_episodes_info.groupby('Season')['Stars'].mean().
    reset_index()
189
190 plt.figure(figsize=(10, 6))
191
192 # Line plot for Stars trend by season
```

```
sns.lineplot(data=season_trend, x='Season', y='Stars', marker='o', color='blue')

# Add titles and labels
plt.title("Stars Trend by Season", fontsize=16)
plt.xlabel("Season", fontsize=14)
plt.ylabel("Average Stars", fontsize=14)

# Show the plot
plt.show()

# Calculate the total Stars by Season
season_totals = all_episodes_info.groupby('Season')['Stars'].mean().reset_index()

plt.figure(figsize=(10, 6))

# Bar plot for average Stars by season
ax = sns.barplot(data=season_totals, x='Season', y='Stars', hue='Season', dodge=False, legend=False)

# Add values to the top of each bar
for container in ax.containers:
    ax.bar_label(container, fmt="%.2f", label_type="edge")  # Show 2 decimal points

# Add titles and labels
plt.title("Average Stars by Season", fontsize=16)
plt.xlabel("Season", fontsize=14)
plt.ylabel("Average Stars", fontsize=14)

# Show the plot
plt.show()

from matplotlib_venn import venn2

# Extract episode titles for top 8 votes and stars
votes_episodes = set(top_8_episodes_votes['Episode_Title'])
stars_episodes = set(top_8_episodes_Stars['Episode_Title'])

# Create the Venn diagram
plt.figure(figsize=(8, 8))
venn = venn2([votes_episodes, stars_episodes], set_labels=('Top 8 by Votes', 'Top 8 by Stars'))


# Title and display
plt.title("Overlap Between Top 8 Episodes by Votes and Stars", fontsize=16)
plt.show()

# Plot a scatter plot to show  Votes trend

plt.figure(figsize=(10, 6))

# Scatter plot for Votes
sns.scatterplot(data=episodes_celebrities_df, x='Episode_Index', y='Votes', color='blue')
```

```
244
245 # Add titles and labels
246 plt.title("Episodes With Guest star Votes trend", fontsize=16)
247 plt.xlabel("Episode_Index", fontsize=14)
248 plt.ylabel("Votes", fontsize=14)
249
250 # Show the plot
251 plt.show()
252 sorted_df = episodes_celebrities_df.sort_values(by='Votes', ascending=
    False)
253 print(sorted_df[['Guest_Star_Name', 'Season','Episode_Number', 'Votes'
    ]].head(2))
254
255 plt.figure(figsize=(10, 6))
256
257 # Scatter plot for Votes
258 sns.scatterplot(data=episodes_celebrities_cheering_df, x='Episode_Index
    ', y='Votes', color='blue')
259
260 # Add titles and labels
261 plt.title("Episodes With Guest star(with audience cheering) Votes trend
    ", fontsize=16)
262 plt.xlabel("Episode_Index", fontsize=14)
263 plt.ylabel("Votes", fontsize=14)
264
265 # Show the plot
266 plt.show()
```

Listing 1: Example Python Code

B Appendix B: Additional Figures

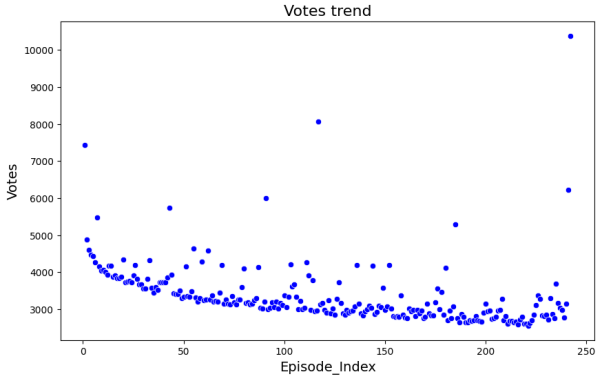

Figure 1: Votes Trends

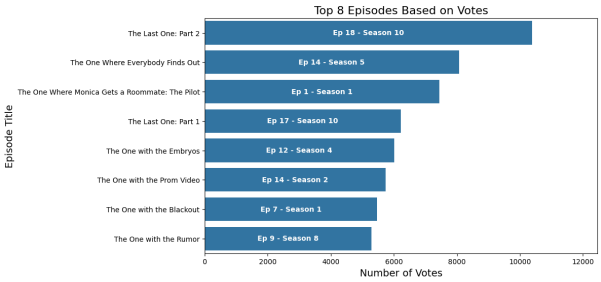

Figure 2: Top 8 Episodes based on votes

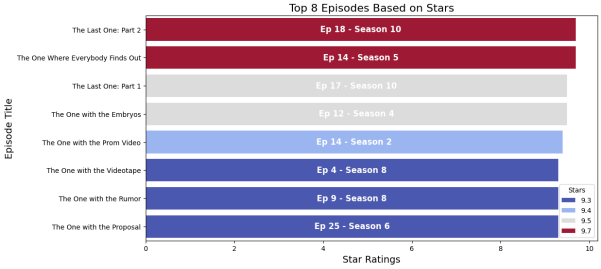

Figure 3: Top 8 Episodes based on IMDb stars

Figure 4: IMDb Stars Trends

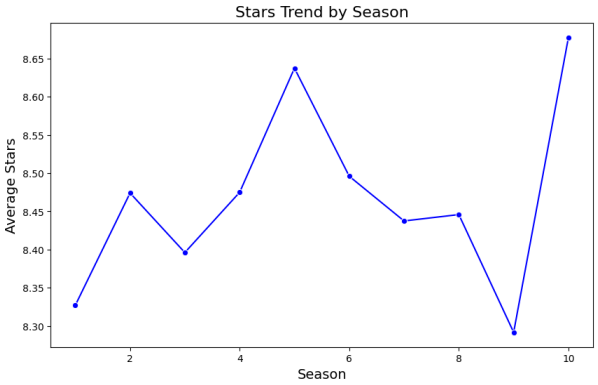

Figure 5: IMDb Stars Trends by Season

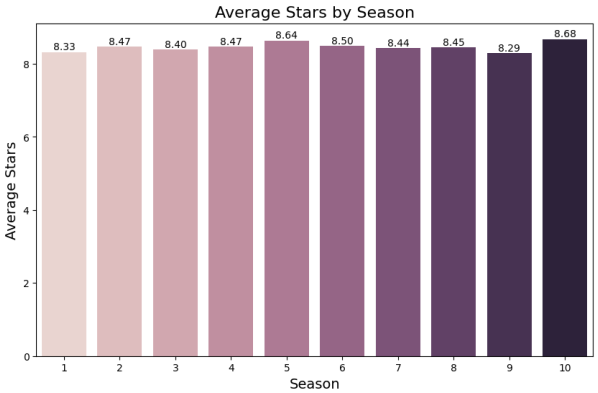

Figure 6: Average Stars by Season

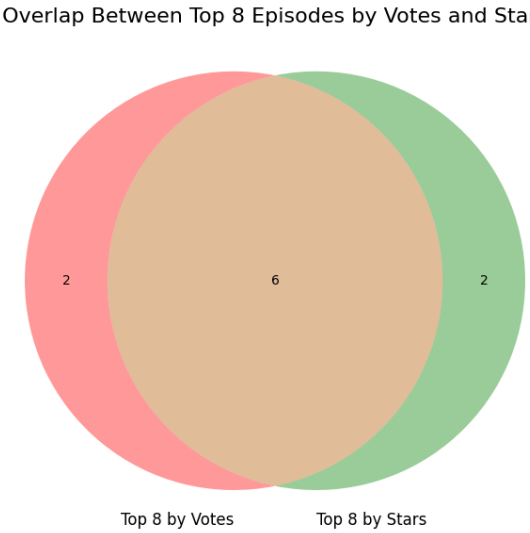

Figure 7: Overlap Between Top 8 Episodes by Votes and Stars

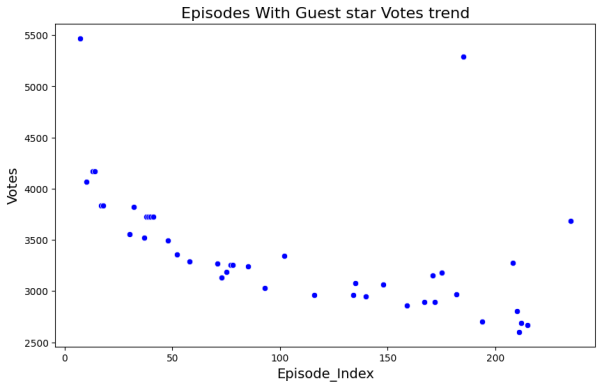

Figure 8: Episodes with Guest Star Votes Trends

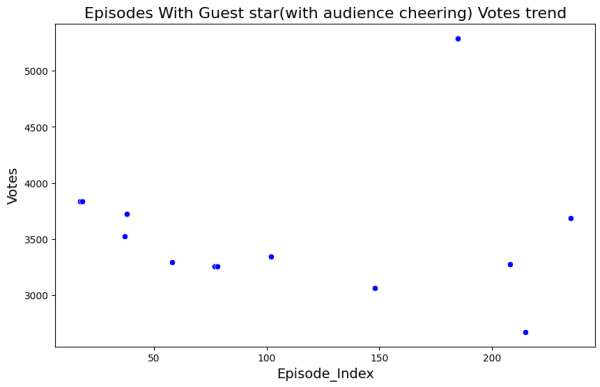

Figure 9: Episodes with Guest Star(with audience cheering) Votes Trends