Analysis of Netflix User Interaction Patterns

Topic 4
PG Group 10
Ali Asgar Baghdadwala, Cyrus Melroy Fernandes
Pavan Kumar, Sharvary Rahatade

ECS757P - Digital Media and Social Networks - 2023-2024

***Abstract*—This paper presents a detailed analysis of user interaction patterns within Netflix. The study includes four main tasks: network construction and statistics, movie popularity analysis, community detection, and customer engagement dynamics. In the first task, we use a small subset of 200 customers and construct a bipartite graph to represent the relationship between customers and movies in terms of the ratings given by the customers to the movies. In the second task, we use a larger subset of 10,000 customers and discover the influential movies that form the foundation of the Netflix experience by deploying network analysis techniques that quantify movie centrality based on weighted sum of views and user ratings. Further exploration into the diffusion patterns of these popular movies gives us an empirical view of user engagement trends and provides crucial insights for curating content and improving recommendations. In the third task, we again use the dataset of 10,000 customers along with movie genres to explore the shared interest of customers towards various genres of movies, where we implement community detection algorithms to find the most influential communities addressing the genres and interests of users. In task 4, we again use the dataset of 10,000 customers along with movie genres to visualize engagement trends, pinpoint peak periods (monthly mean weights), and explore factors that keep users engaged (correlation analysis).**

II. Introduction

In the current era of digital entertainment, Netflix has emerged as a leader, redefining how individuals consume media content. With its vast library of movies, TV shows, and original series, Netflix is not only a source of entertainment but also offers an opportunity to understand user interaction patterns, which is crucial for developers to optimize content delivery and enhance user experience. However, it becomes challenging to analyse this data effectively due to the dynamic nature of user behaviours and wide range of content options.

In this project, we study and apply graph networks using Python to understand user interactions and our results provide insights that can benefit both content creators and platform administrators

A. Research Problem and Motivation

In Task 1, the research problem involves understanding customer preferences and behavior in terms of movie ratings. The motivation behind this task is the importance of having personalized recommendations. Critical analysis of customer-movie relationships such as ratings allow us to unveil patterns and trends for developing effective recommendation systems.

In Task 2, our main aim was to identify influential movies on Netflix and understand how their popularity is spread over the ecosystem. The motivation of the research was based on improving the content recommendation algorithms which will lead to better overall user satisfaction and retention. To do so we identified the diffusion patterns in the most popular content so as to provide more dynamic and personalised recommendations.

Task 3 is motivated by the need to gather crucial data on user interactions across various timeframes and their shared characteristics, which will be essential in understanding user engagement in watching patterns, and enhancing user experience with better understanding of their interests which can significantly improve recommendation systems on entertainment platforms like Netflix. This can lead to better suggestion of movies and genres which users are likely to watch and enjoy the most.

In Task 4, the research problem involves analyzing customer engagement dynamics, identifying peak interaction periods, exploring factors influencing sustained engagement, and observing impact of trending genres.

1) Customer Engagement Dynamics:

A number of factors, including the availability of content, may cause users' tastes to change over time. In order to give Netflix a competitive edge, it is imperative to investigate these trends and determine what kinds of content the majority of users watch. This enables online platforms such as Netflix to anticipate user behaviour and modify the content available on their platform appropriately.

2) Peak Interaction Periods:

Digital media companies need to know when users are most active on their platforms - the peak interaction time. The best times to interact could be over the weekend, during a holiday, or during a well-liked movie or web series. This analysis will help Netflix increase the quantity of new content it offers on its platform and more effectively target its audience. Factors Influencing Sustained Engagement.

3) Factors Influencing Sustained Engagement:

A platform's level of success is determined by its capacity to maintain user interest. Through the analysis of user-friendly interfaces, personalised recommendations, and engaging content that keeps users coming back, platforms gain valuable insights into consumer behaviour. The platform's

long-term success depends on you building a devoted user base and increasing retention rates, both of which can be achieved by fixing these issues.

4) Impact of Trending Genres:

Data is a crucial element in sustaining user engagement. Platforms can make more informed decisions about what to purchase and recommend if they are aware of the current trends in film genres. Platforms can ascertain what truly resonates with viewers by examining the ways in which these well-liked genres impact user interaction. This data is a veritable gold mine for tailoring content libraries to preserve user satisfaction. Beyond satisfied viewers, though, genre trends also inform marketing strategies that help platforms attract new customers and maintain their position in the very competitive streaming market.

B. Challenges to Address

The main challenge encountered in Task 1 is visualization of the bipartite network of the entire dataset of 200 customers. Visualization is the key to conduct statistical analysis and infer insights from data. However, a bipartite network of 200 customer nodes with multiple edges connecting each node is practically impossible to visualize and infer due to severe label crowding, making it impossible to identify the nodes at the ends of an edge. Hence, for visualization purpose, we extract a random sample of 30 nodes and plot the graph.

The main objective of Task 2 was to specifically identify and pinpoint extremely popular movies by using either the number of views or ratings as a metric. However, we have chosen to integrate both to construct a weighted sum that considers both the quantity of interactions (views) and the quality of user satisfaction (ratings). This innovative method required really careful consideration to really ensure, neither of the metric disproportionately influenced the overall popularity score thus presenting a very nuanced challenge of determining the optimal weighting scheme to accurately reflect a movie's true popularity.

The core challenges in Task 3 are to find the watching patterns based on shared interests, movie ratings, genres that customers are watching most of the time. However, the dataset for movies and users is very huge and there are multiple genres linked to each movie. There are several filters to apply on the merged dataset of users and genres they are interested in and number of times they have watched the same genre of movies, however when we map the movies and genres, there are 100k data points that can be created but based on the filters, we should be able to shortlist the most important data points between edge and nodes to apply an efficient community detection algorithm.

Cleaning and preprocessing data was a preliminary challenge in Task 4. The dataset containing the genres was majorly empty and the dataset containing the dates (used to find customer engagement over peak interaction period) had a formatting issue. The 'Date' column was not compatible with the said format. This was solved by using appropriate format parameters in the pd.to_datetime() function. Another issue was merging two huge datasets and identifying which columns could be used as primary keys/identifiers. It was of paramount importance to make sure that no data was lost while merging two datasets. Visualizing the trending genres in the form of a pie chart was yet another challenge. The pie chart made using the entire dataset containing all genres was not legible and hence it was impossible to interpret the most trending genres and its impacts. Hence, we use only the top 15 trending genres for the pie chart visualization.

III. RELLATED WORK

There has been substantial interest in the field of video-on-demand (VOD) services with the growth of platforms like Netflix. Various analytical and algorithmic approaches have been used in recent years to enhance user engagement and content recommendation.

Studies on how machine learning methods can enhance content recommendation include a widely cited paper "The Netflix Recommender System: Algorithms, Business Value, and Innovation" on the Netflix Prize dataset by Gomez-Uribe and Hunt (2015) [1], which features complex algorithms to combine user data with content features to improve the accuracy of movie recommendations. Through this approach, customers can expect a more personalized content delivery.

Important attention has also focused on network-based analytics due to their capacity to understand complex user interaction patterns and content flows. For instance, the study " Maximizing Aggregate Recommendation Diversity: A Graph-Theoretic Approach (2011) [2] shows the application of graph theory in mapping the complex interactions between users and content, using network metrics to predict user behavior and content popularity. In this study, complex network metrics like centrality and clustering coefficients are used to understand how users interact with content and how these interactions influence content popularity. This approach shows how structural analysis can be used in order to understand and predict user preferences

In the article, "Social Network Comparison of Netflix, Disney+, and OCN on Twitter Using NodeXL" (2022) [3], the authors analyze the influence of social media on content popularity. This study shows the role of social media platforms like Twitter in determining content success and user interest. It uses NodeXL to see and understand how people interact with content on social media.

In the study "Recommendations and Results Organization in Netflix Search" (2021) [4], Lamkhede and Kofler show how integrating search algorithms with user interface can enhance user experience. It shows how Netflix arranges search results and suggestions to meet different user needs.

In the paper "Building a Movie Recommendation System Using Neo4j Graph Database" (2023) [5], the authors discuss the application of the Neo4j graph database in developing recommendation systems. They developed a movie recommendation algorithm using the k-NN similarity algorithm and FastRP node embedding machine learning. Using this approach, relationships between movies can be modeled based on various attributes like actors and genres, which eventually enhances recommendation systems.

In the article "The structure and function of complex networks" [6] by M. E. J. Newman, the author discusses about the structure

and function of complex networks. In recent years, researchers have developed techniques to understand these systems. The article reviews concepts such as small-world effects, degree distributions, clustering, network correlations, random graph models, and models of network growth.

IV. Dataset and network presentation

The dataset for this project is extracted from a larger 'Netflix Prize data' dataset [7] hosted on Kaggle, an online Data Science community that hosts datasets and competitions. The dataset contains customer interaction data in the form of movie views and ratings recorded at a certain data. Two subsets have been generated: a smaller subset comprising data from 200 customers for Task 1, and a larger subset comprising data from 10,000 customers for Task 2, 3, and 4.

The node list file for each of the 200 and 10,000 customers contain the following attributes:

- ID: node identifier (C – Customer, M – Movie) (String)
- Label: node label (String)
- Type: node type (Customer or Movie) (String)

The edge list file for each of the 200 and 10,000 customers contain the following attributes:

- Source: Customer ID (String)
- Target: Movie ID (String)
- Weight: Customer rating for the particular movie (Integer)
- Date: Date (MM-DD-YYYY format) at which the rating was given (String)

The file "movie_titles.csv" contains the following attributes:

- ID: Movie ID (Integer)
- Year: Year of release of the movie (Integer)
- Title: The name of the movie (String)

For tasks 3 & 4, the dataset has been expanded to include the movie genre information provided by the netflix-prize-with-genres GitHub repository [8]. A new column 'MovieGenre' is added to the node list file of 10,000 customers and a separate file is made. The new file consists of all columns in the original file along with the new column representing genre of each movie for the movie nodes. This field is kept empty for customer nodes

A. Task 1

In Task 1, we use the subset of 200 customers from the edge list file to construct a bipartite graph network. In a bipartite graph there are 2 sets of nodes and every edge connects a node from one set to other, and no edges exist between nodes within the same set. In this task, the bipartite graph represents the relationship between customers and movies in terms of the ratings given by the customers to the movies. For this task, we use the following Python libraries: NetworkX, Matplotlib, Pandas

1) Network Construction

An edge list is read from the CSV file containing the following attributes – Source (Customer), Target (Movie), Weight (Rating given by the customer), Date (date at which the rating was given). The code then initializes a NetworkX graph object named 'G' as an undirected graph. Each node in the graph is assigned to either the customers set (0) or the movies set (1). Edges in the graph are color-coded based on the weight attribute, which corresponds to the rating given by the customers. To achieve this, a dictionary is created, mapping different rating values to specific colors. A legend is included in the graph to map each color to its corresponding rating.

2) Network Statistics

a) Clustering Coefficient: 0

This indicates that there are no triplets of nodes that are connected, which is expected in a bipartite network since no two nodes of the same set are directly connected to each other.

b) Maximum Node Betweenness Centrality: 0.153095

c) Average Betweenness: 0.00162753

Nodes with high betweenness centrality act as bridges within the network, suggesting that certain users or movies play a pivotal role in connecting disparate parts of the user-movie graph. These nodes could be key to spreading new trends or recommendations through the network.

d) Maximum Degree: 299

e) Node with Maximum Degree': 'C769'

Identifying the node with the maximum degree helps pinpoint the most engaged user or the most popular movie within the subset. This indicates potential trends and preferences within the Netflix ecosystem, revealing what types of content drive the most engagement

TABLE I : STATISTICS OF THE 5 MOST INFLUENTIAL NODES SORTED BY DEGREE DISTRIBUTION

Node	Degree	Betweenness Centrality
C769	299	0.153095
C79	197	0.059801
C7	195	0.047093
C134	169	0.041451
C481	167	0.043964

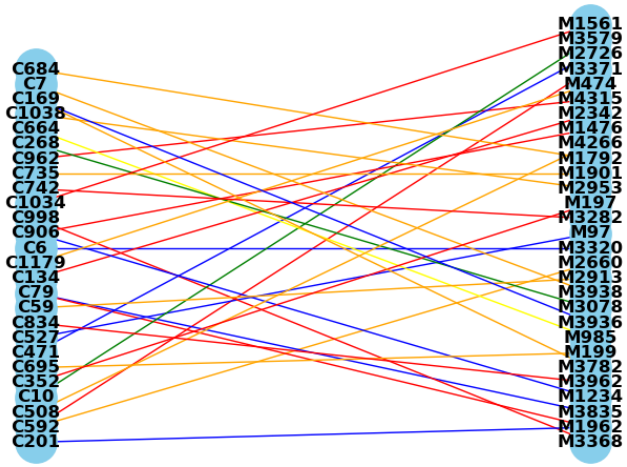

Fig 1: Visualization of the bipartite network constructed using 30 samples from the dataset. Each edge represents the rating given by the customer to the movie. Edges are color-coded to map ratings from 1-5

V. NETWORK ANALYSIS METHODOLOGY

A. *Task 2*

In Task 2, we looked at a large dataset of 10,000 users to assess Netflix movie popularity. To depict the interactions between users and moives, we built a bipartite graph using the NetworkX library. By calculating the weighted sum of views and ratings, we created a composite metric that offers a more nuanced understanding of popularity by capturing both the amount of viewership and the degree of user satisfaction. These two dimensions were combined in our methodological framework to create a comprehensive indicator of a movie's influence within the network. With the help of this method, it was possible to find films that had both a large number of views and a high rating from users, which suggested that the user engagement levels were above average. Our goal was to highlight weighted interactions in order to highlight subtle aspects of content popularity that may go unnoticed by traditional analysis methods that merely count views. Because it makes sure that both highly regarded and widely viewed films are taken into account when developing content curation strategies, this analysis is essential for improving recommendation systems and laying the groundwork for better user experiences. The knowledge gained from this study may help Netflix develop a fairer recommendation algorithm that promotes a varied range of content in addition to popular titles.

B. *Task 3*

After analysing the customer data, we only consider the top-rated movies of each user i.e., rating=5, and after discovering majority of datapoints lie in 2005, where there was highest user engagement, we were able to understand if the same users have watched and liked the same genres in the same time period, which will give us the best insights on users watching patterns. Then we map the users in this filtered data with movie genres. Since there are multiple genres linked to each movie, we encode these combinations into single integer numbers which correspond to the movie combinations using python library LabelEncoder from sklearn.preprocessing. Next, we implement data cleaning methods such as removing null values and duplicate entries after merging the data.

We also consider the frequency of each genre combination watched by each user which we consider as the weight parameter to apply on community detection algorithms. After trying several methods, we find that hierarchical clustering will work best as it calculates the Euclidian distance with each genre and their weights. We also use StandardScalar python package to normalize the weights for each user to see how close they are by using linkage method.

C. *Task 4*

We use the node and edge files of 10,000 customers dataset along with movie genres. The edge list dataset represents relationship between entities (e.g., users and movie genres), while the node list dataset contains attributes of nodes (e.g., genre names). We merge the above two datasets to identify peak interaction periods, which we depict using a time series line plot. We aggregate on monthly level to find the peak interaction over the time series data. We also find which genres are most watched in those peak interaction periods.

VI. RESULTS AND DISCUSSION

A. *Task 2*

A distinct ranking of movie popularity based on viewership and ratings is provided by the Netflix dataset. 'M1905's incredible impact suggests that it has a broad appeal and could be essential for content strategy and audience engagement. Beyond this front-runner, popular films like "M4306" and "M2452" highlight how crucial it is to accommodate a range of viewer interests and show off the diversity of content available on the platform. Despite having comparable viewership and ratings, these top films have very different influence scores, highlighting the small elements that go into a Netflix film's success. Netflix user data diffusion patterns show the presence of a subset of "super users" whose behaviour greatly influences the platform's trends. An additional temporal analysis reveals the peaks and troughs in the box office success of films; the peak of views for 'M2874' highlights the significance of deliberate content promotion and broader societal influences. The instability of viewership trends necessitates the need for Netflix to have an agile analytics framework to promptly modify marketing strategies and suggestions. The value of real-time data analysis in the streaming domain is underscored by the opportunity to leverage these shifts to personalise the viewing experience and improve user engagement.

TABLE II: TOP 10 INFLUENTIAL MOVIES BASED ON WEIGHTED RATINGS AND VIEWS

Movie ID	Views	Total Weighted Rating	Influence Score
M1905	4147	17280	71660160
M4306	3288	14249	46850712
M2452	3210	14262	45781020
M571	3304	13156	43467424
M3938	3225	13399	43211775
M4432	3358	12655	42495490
M2152	3476	11796	41002896
M3860	3428	11777	40371556
M3962	2998	13217	39624566
M2782	2910	12514	36415740

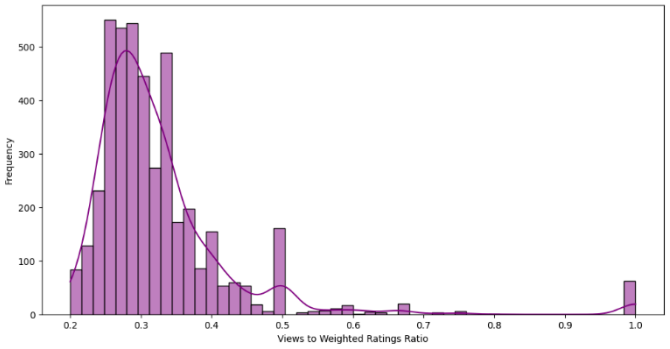

Fig 2: Distribution of views to weighted ratings ratio

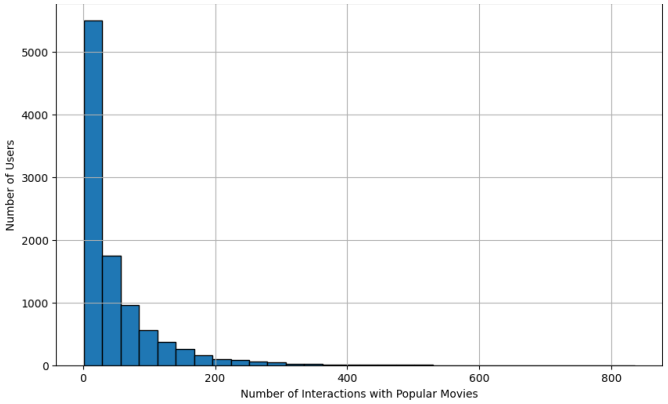

Fig 3: Distribution of interactions with popular movies among users

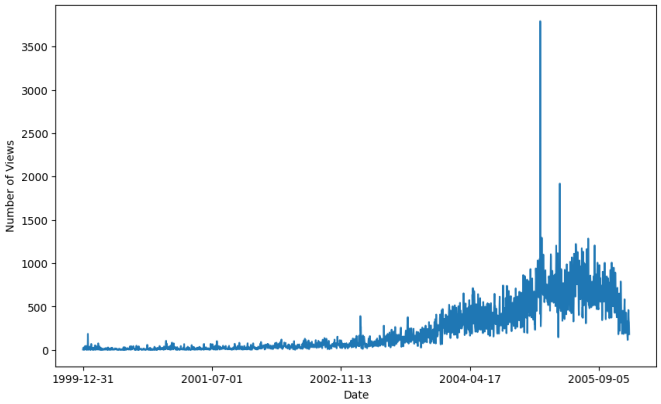

Fig 4: Temporal distribution of popular movie views

B. *Task 3*

We find that there are two major communities in clusters, each of these clusters having top 5 popular genre combinations based on highest number of views and highest ratings. We discover that one cluster is around major genres such as 'Action' followed by 'adventure', 'fantasy', 'thriller', 'crime'. The second cluster consists of 'Drama' as a majority genre followed by 'comedy', 'romance', 'mystery'. We can see that 'Drama' community is slightly having more popularity among viewers compared to 'Action' community. This would suggest that there is more scope to recommend and add genres from community two and also improve the content for community one in order to gain more viewership from action movies.

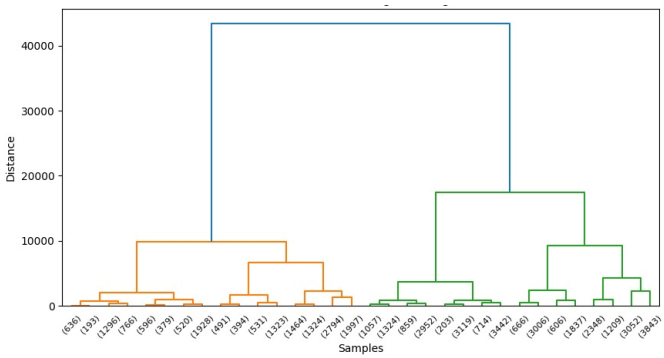

Fig 5: Community detection algorithm using Hierarchical Clustering

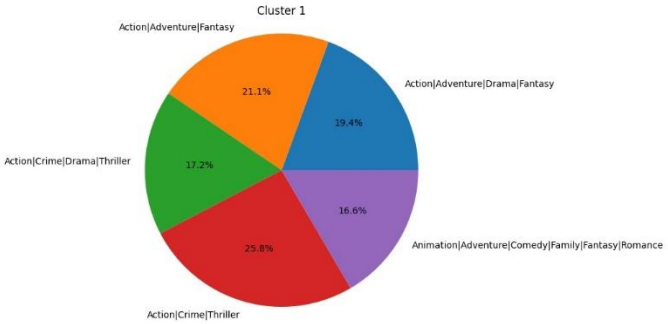

Fig 6: Cluster 1 Genre combinations

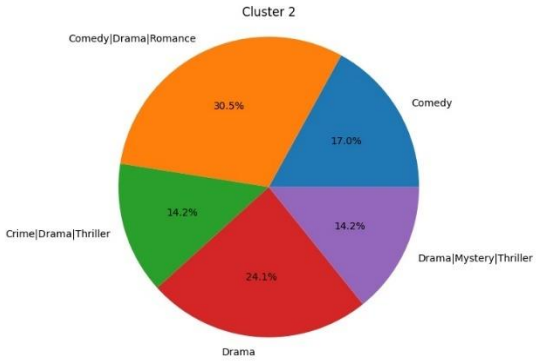

Fig 7: Cluster 2 Genre combinations

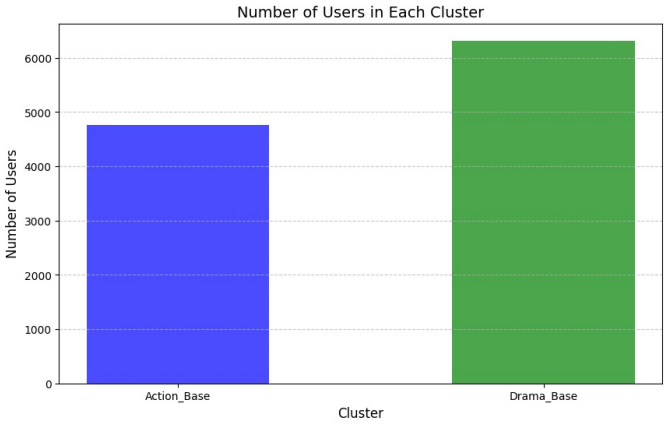

Fig 8: Cluster size comparison

C. *Task 4*

We merge the edge list and movie genres datasets of 10,000 customers and aggregate it on year and month. We obtain the year and month integers using pd.dt.year and pd.dt.month python packages respectively. The resulting dataset displays the aggregated interaction count on monthly level along each year. We generate a time series line plot of this data to identify the peak interaction months in terms of the counts. It is observed that the highest interaction period was in the year 2005 with the most watched genres being 'Drama' and 'Comedy'. The highest viewership was in the month of June with the count being 26354 followed by July with a count of 25481. The highest watched genre is 'Drama', which was watched 524 times in the month of March 2005. In the same month, 'Comedy' genre was watched 460 times. In conclusion, these are the genres that yielded the highest viewership during the peak interaction period.

TABLE III: PEAK INTERACTION MONTHS

Year	Month	Count
2005	June	26354
2005	July	25481
2005	January	25117
2005	August	24748
2005	May	23767

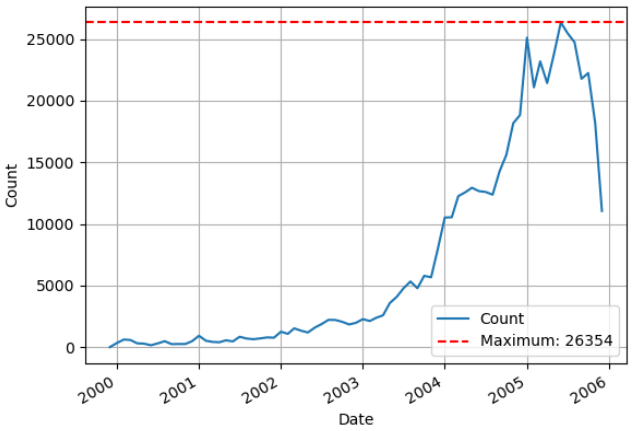

Fig 9: Temporal peak interaction data

TABLE IV: HIGHEST WATCHED GENRES IN THE PEAK INTERACTION PERIOD

Year	Month	Genre	Count
2005	3	Drama	524
2005	6	Drama	518
2005	3	Comedy	460
2005	11	Drama	457
2005	7	Drama	454

VII. CONCLUSIONS AND PERSPECTIVES

In this project, we analyzed Netflix user interaction patterns through network analysis and community detection algorithms, that allow us to gain insights into developing effective recommendation systems and enhance viewing experience. For future work, we can predict future trends based on user viewing patterns and improve user experience, and provide better recommendation systems based on community detection algorithms. We can also extend our analysis to other entertainment platforms beyond Netflix.

VIII. References

[1] C. A. G. U. a. N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," *ACM Transactions on Management Information Systems,* vol. 6, no. 4, pp. 1-19, 2015.

[2] Y. K. Gediminas Adomavicius, "Maximizing Aggregate Recommendation Diversity: A Graph-Theoretic Approach," in *ACM International Conference*, Chicago, 2011.

[3] K. S. W. B. J. C. Soochang Lee, "Social Network Comparison of Netflix, Disney+, and OCN on Twitter," *International Journal of Advanced Culture Technology,* vol. 10, no. 1, pp. 47-54, 2022.

[4] C. K. Sudarshan Lamkhede, "Recommendations and Results Organization in Netflix Search," in *RecSys '21: Proceedings of the 15th ACM Conference on Recommender Systems*, Amsterdam, 2021.

[5] A. H. Izdihar, N. D. Tsaniyah, F. Nurdini, B. R. Mufidah and N. A. Rakhmawati, "Building a Movie Recommendation System Using Neo4j Graph Database: A Case Study of Netflix Movie Dataset," in *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, Manama, 2024.

[6] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review,* vol. 45, no. 2, pp. 167-256, 2003.

[7] Netflix, Chris Crawford, "Netflix Prize data - Dataset from Netflix's competition to improve their reccommendation algorithm," 13 11 2019. [Online]. Available: https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data/data. [Accessed 17 04 2024].

[8] T. Carraro, "tommasocarraro / netflix-prize-with-genres," 29 4 2020. [Online]. Available: https://github.com/tommasocarraro/netflix-prize-with-genres/blob/master/netflix_genres.csv. [Accessed 17 4 2024].

IX. APPENDIX

A. Task 1

```
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv('topic4_edge_list_200_customers.csv').sample(30).to_numpy()

G = nx.Graph()  # Creating bipartite graph object

for source, target, weight, date in data:  # Adding nodes and edges to the graph
    G.add_node(source, bipartite=0)
    G.add_node(target, bipartite=1)
    G.add_edge(source, target, weight=weight, date=date)

#Dictionary mapping ratings to their color codes
weight_colors = {
    1: 'yellow',
    2: 'green',
    3: 'blue',
    4: 'orange',
    5: 'red',
}

legend_labels = {rating: plt.Line2D([0], [0], marker='o', color=color, label=rating) for rating, color in weight_colors.items()}
#legend box

# Separating nodes by bipartite sets
top_nodes = {n for n, d in G.nodes(data=True) if d['bipartite'] == 0} # customer nodes
bottom_nodes = set(G) - top_nodes    # movie nodes

# Plotting the bipartite graph with different color edges for each weight category
pos = {node: (1, i) for i, node in enumerate(top_nodes)}  # Fixed position for top nodes
pos.update({node: (2, i) for i, node in enumerate(bottom_nodes)})  # Fixed position for bottom nodes

edge_colors = [weight_colors[G[u][v]['weight']] for u, v in G.edges()]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_color='skyblue', node_size=800, edge_color=edge_colors)
plt.legend(handles=legend_labels.values(), title='Ratings', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.savefig("bipartite_sample.png")
plt.show()

data = pd.read_csv('topic4_edge_list_200_customers.csv').to_numpy()

G = nx.Graph()

for source, target, weight, date in data:
    G.add_node(source, bipartite=0)
    G.add_node(target, bipartite=1)
    G.add_edge(source, target, weight=weight, date=date)

top_nodes = {n for n, d in G.nodes(data=True) if d['bipartite'] == 0}
bottom_nodes = set(G) - top_nodes

pos = {node: (1, i) for i, node in enumerate(top_nodes)}
```

```
pos.update({node: (2, i) for i, node in enumerate(bottom_nodes)})

edge_colors = [weight_colors[G[u][v]['weight']] for u, v in G.edges()]
nx.draw(G, pos, with_labels=True, font_weight='bold', node_color='skyblue', node_size=800, edge_color=edge_colors)
plt.legend(handles=legend_labels.values(), title='Ratings', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.savefig("bipartite_full_200.png")
plt.show()

data = pd.read_csv('topic4_edge_list_200_customers.csv').to_numpy()
G = nx.Graph()

for source, target, weight, date in data:
    G.add_node(source, bipartite=0)
    G.add_node(target, bipartite=1)
    G.add_edge(source, target, weight=weight, date=date)

# Network statistics
clustering_coefficient = nx.average_clustering(G)
degree_distribution = dict(G.degree())
max_degree = max(degree_distribution.values())
betweenness_centrality = nx.betweenness_centrality(G)
average_betweenness = sum(betweenness_centrality.values()) / len(betweenness_centrality)

# Display results
print(f"Clustering Coefficient: {clustering_coefficient}")
print("\n")
print(f"Degree Distribution: {degree_distribution}")
print("\n")
print(f"Maximum Degree: {max_degree}")
print("\n")
print(f"Betweenness Centrality: {betweenness_centrality}")
print("\n")
print(f"Average Betweenness: {average_betweenness}")

l = list(degree_distribution.items())
l.sort(key=lambda x: x[1], reverse=True)
l

l2 = list(betweenness_centrality.items())
l2.sort(key=lambda x: x[1], reverse=True)
l2
```

B. Task 2

```
import pandas as pd

# Load the datasets
edges_df = pd.read_csv('topic4_edge_list_10000_customers.csv')
nodes_df = pd.read_csv('topic4_node_list_10000_customers.csv')

import networkx as nx

# Initialize the graph
G = nx.Graph()

# Add movie and customer nodes
G.add_nodes_from(nodes_df[nodes_df['Type'] == 'Movie']['ID'], bipartite=1)
G.add_nodes_from(nodes_df[nodes_df['Type'] == 'Customer']['ID'], bipartite=0)

# Add edges with weights (ratings)
for _, row in edges_df.iterrows():
    G.add_edge(row['Source'], row['Target'], weight=row['Weight'])

# This will create a dictionary with movies as keys and their weighted degree as values
movie_degrees = {node: val for node, val in G.degree(weight='weight') if isinstance(node, str) and node.startswith('M')}

# Sort the movies based on the weighted degree
sorted_movies = sorted(movie_degrees.items(), key=lambda x: x[1], reverse=True)

# Get the top 10 influential movies
top_10_movies = sorted_movies[:10]
for movie, degree in top_10_movies:
    print(f"Movie ID: {movie}, Weighted Degree: {degree}")

import pandas as pd

# Load the datasets
edges_df = pd.read_csv('topic4_edge_list_10000_customers.csv')

# Calculate the weighted sum of ratings and views for each movie
# First, multiply the rating (weight) by the number of views for each movie
edges_df['weighted_rating'] = edges_df['Weight']

# Group by movie and sum up the weighted ratings
movie_influence = edges_df.groupby('Target')['weighted_rating'].sum()

# Now, to get the views per movie, we simply count the number of occurrences of each movie in the edges dataset
movie_views = edges_df['Target'].value_counts()

# Combine both views and weighted ratings into a single DataFrame
movie_influence_df = pd.DataFrame({
    'views': movie_views,
    'total_weighted_rating': movie_influence
})

# Calculate the final influence score by multiplying the views by the average rating for each movie
movie_influence_df['influence_score'] = movie_influence_df['views'] * movie_influence_df['total_weighted_rating']

# Sort the movies by their influence score
ranked_movies = movie_influence_df.sort_values(by='influence_score', ascending=False)
```

```
# Save the DataFrame to a CSV file
ranked_movies.to_csv('influence_scores.csv', index_label='MovieID')

# Display the top 10 influential movies based on the calculated score
ranked_movies.head(10)

import matplotlib.pyplot as plt

# Plotting the top 10 influential movies
top_10_movies = ranked_movies.head(10)
top_10_movies['influence_score'].plot(kind='bar', figsize=(10, 6), color='skyblue')
plt.title('Top 10 Influential Movies Based on Weighted Ratings and Views')
plt.xlabel('Movie ID')
plt.ylabel('Influence Score')
plt.xticks(rotation=45)
plt.show()

import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(data=movie_influence_df, x='views', y='total_weighted_rating', size='influence_score', legend=False, sizes=(20,
200))
plt.title('Scatter Plot of Views vs. Weighted Ratings')
plt.xlabel('Views')
plt.ylabel('Total Weighted Ratings')
plt.grid(True)
plt.show()

# Binning data for heatmap
movie_influence_df['views_bin'] = pd.qcut(movie_influence_df['views'], 5, duplicates='drop')
movie_influence_df['ratings_bin'] = pd.qcut(movie_influence_df['total_weighted_rating'], 5, duplicates='drop')

# Creating pivot table for heatmap
heatmap_data = movie_influence_df.pivot_table(index='views_bin', columns='ratings_bin', values='influence_score',
aggfunc='mean')

# Plotting the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_data, cmap='viridis', annot=True)
plt.title('Heatmap of Movies by Views and Ratings')
plt.xlabel('Total Weighted Ratings Bins')
plt.ylabel('Views Bins')
plt.show()

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

# Load the movie influence data
movie_influence_df = pd.read_csv('influence_scores.csv')

# Calculate the ratio of views to weighted ratings
movie_influence_df['views_to_weight_ratio'] = movie_influence_df['views'] / movie_influence_df['total_weighted_rating']

# Plotting the distribution of the views to weighted ratings ratio
plt.figure(figsize=(12, 6))
```

```
sns.histplot(movie_influence_df['views_to_weight_ratio'], bins=50, kde=True, color='purple')
plt.title('Distribution of Views to Weighted Ratings Ratio')
plt.xlabel('Views to Weighted Ratings Ratio')
plt.ylabel('Frequency')
plt.show()

import pandas as pd
import matplotlib.pyplot as plt

# Assuming influence_scores.csv has already been saved from part a
movie_influence_df = pd.read_csv('influence_scores.csv')

# Define a threshold to identify popular movies
threshold = movie_influence_df['influence_score'].quantile(0.75)
popular_movies = movie_influence_df[movie_influence_df['influence_score'] >= threshold]['MovieID']

# Load the edge dataset to analyze interactions
edges_df = pd.read_csv('topic4_edge_list_10000_customers.csv')

# Filter interactions to only those involving popular movies
popular_interactions = edges_df[edges_df['Target'].isin(popular_movies)]

# Analyze how many users have interacted with popular movies
user_interactions = popular_interactions.groupby('Source').size()

# Plot the distribution of interactions per user
plt.figure(figsize=(10, 6))
user_interactions.hist(bins=30, edgecolor='black')
plt.title('Distribution of Interactions with Popular Movies Among Users')
plt.xlabel('Number of Interactions with Popular Movies')
plt.ylabel('Number of Users')
plt.show()

# Optionally, investigate the temporal aspect if dates are available
if 'Date' in edges_df.columns:
    edges_df['Date'] = pd.to_datetime(edges_df['Date'])
    temporal_distribution = popular_interactions.groupby('Date').size()
    plt.figure(figsize=(10, 6))
    temporal_distribution.plot()
    plt.title('Temporal Distribution of Popular Movie Views')
    plt.xlabel('Date')
    plt.ylabel('Number of Views')
    plt.show()

# Importing pandas to handle data processing
import pandas as pd
# Load the edge dataset to analyze interactions
edges_df = pd.read_csv('topic4_edge_list_10000_customers.csv')
# Convert 'Date' column to datetime
edges_df['Date'] = pd.to_datetime(edges_df['Date'])
# Find the date with the peak number of views
peak_date = edges_df.groupby('Date').size().idxmax()
# Filter the interactions for the peak date
peak_day_interactions = edges_df[edges_df['Date'] == peak_date]
# Find the movie with the most interactions on the peak daytop_movie_id_on_peak_day =
peak_day_interactions.groupby('Target').size().idxmax()
top_movie_id_on_peak_day
```

C. Task 3

```
import zipfile
zip_file_path = "Topic4_Netflix_dataset.zip"  # Replace with your actual path

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
  zip_ref.extractall()  # Extracts all files in the zip to the current directory

import pandas as pd

df_cust = pd.read_csv('/content/Topic4_Netflix_dataset/topic4_edge_list_10000_customers.csv')

df_cust

filtered_df = df_cust[ (df_cust['Weight'] == 5) ]


import pandas as pd

# Assuming 'Date' is the column containing dates
filtered_df['Date'] = pd.to_datetime(filtered_df['Date'])

# Filter to keep only dates from 2005
filtered_df = filtered_df[filtered_df['Date'].dt.year == 2005]

# Display the filtered DataFrame
print(filtered_df)


filtered_df['Date'].value_counts()

df_cust = filtered_df[['Source','Target']]

df_cust.dropna(inplace=True)

df_cust

df_movie= pd.read_csv('/content/Topic4_Netflix_dataset/topic4_node_list_genre_10000_customers.csv')

df_movie.dropna(inplace=True)

df_movie

#df_cust.merge(df_movie,on='Label')

df_file = df_cust.merge(
    df_movie, left_on="Target", right_on="Label")

df_file

users = df_file[['Source','MovieGenre']]

from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
label_encoder = LabelEncoder()
```

```
# Fit and transform the "MovieGenre" column
users["MovieGenreEncoded"] = label_encoder.fit_transform(users["MovieGenre"])

print(users)


#users = users[["Source","MovieGenreEncoded"]]

users

users = users.groupby(['Source', 'MovieGenreEncoded']).size().reset_index(name='Count')

# Display the DataFrame with user, genre, and count columns
print(users)

users.drop_duplicates(inplace=True)

users.set_index('Source',inplace=True)

users

from sklearn.preprocessing import OneHotEncoder, StandardScaler

scaler = StandardScaler()
users['normalised'] = scaler.fit_transform(users[['Count']])


users['normalised']

# Plot dendrogram

import pandas as pd
from scipy.cluster import hierarchy
import matplotlib.pyplot as plt
import sys

distance_matrix = users[['MovieGenreEncoded', 'normalised']].values

# Perform hierarchical clustering
linkage = hierarchy.linkage(distance_matrix, method='ward')



# Plot dendrogram
plt.figure(figsize=(10, 5))
dn = hierarchy.dendrogram(linkage,truncate_mode='level', p=4)
plt.title("Hierarchical Clustering Dendrogram")
plt.xlabel("Samples")
plt.ylabel("Distance")
plt.show()


# Assign cluster labels to the original DataFrame
users['Cluster'] = hierarchy.fcluster(linkage, 2, criterion='maxclust')

print(users)
```

```
cluster_user_genres = users.groupby(['Cluster', 'Source'])['MovieGenreEncoded'].apply(lambda x: ', '.join(map(str,
x.unique()))).reset_index(name='MovieGenres')

# Display the cluster user genres
print(cluster_user_genres)

decode_mapper = df_file[['Source','MovieGenre']]
# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the "MovieGenre" column
decode_mapper["MovieGenreEncoded"] = label_encoder.fit_transform(decode_mapper["MovieGenre"])

genre_mapping_dict = dict(zip(decode_mapper['MovieGenreEncoded'], decode_mapper['MovieGenre']))

# Decode the MovieGenreEncoded column
users['DecodedGenre'] = users['MovieGenreEncoded'].map(genre_mapping_dict)

# Display the DataFrame with the decoded genre column
print(users)

import matplotlib.pyplot as plt

# Group by cluster and genre, and count the occurrences
cluster_genre_counts = users.groupby(['Cluster', 'DecodedGenre']).size().reset_index(name='Count')

# Sort genres within each cluster based on counts
cluster_genre_counts['Rank'] = cluster_genre_counts.groupby('Cluster')['Count'].rank(ascending=False, method='dense')

# Select top 5 genres for each cluster
top5_genres = cluster_genre_counts[cluster_genre_counts['Rank'] <= 5]

# Group by cluster and aggregate top 5 genres
top5_genres_per_cluster = top5_genres.groupby('Cluster')['DecodedGenre'].apply(lambda x: ',
'.join(x)).reset_index(name='TopGenres')

# Iterate over each cluster and create a separate figure for each pie chart
for i, cluster in enumerate(top5_genres_per_cluster['Cluster']):
    # Create a new figure for each pie chart
    plt.figure(figsize=(8, 6))

    # Get the top genres and their counts for the current cluster
    labels = top5_genres_per_cluster.loc[top5_genres_per_cluster['Cluster'] == cluster, 'TopGenres'].values[0].split(', ')
    sizes = [top5_genres[top5_genres['Cluster'] == cluster].loc[top5_genres['DecodedGenre'] == genre, 'Count'].values[0] for genre
in labels]

    # Create the pie chart
    plt.pie(sizes, labels=labels, autopct='%1.1f%%')
    plt.title(f'Cluster {cluster}')
    plt.axis('equal')

    # Show the pie chart
    plt.show()



import matplotlib.pyplot as plt
```

```
# Calculate the number of users in each cluster
cluster_sizes = cluster_user_genres.groupby('Cluster').size()


colors = ['blue', 'green']  # Add more colors if needed

# Define names for each cluster
cluster_names = ['Action_Base', 'Drama_Base', ]  # Add more names if needed

# Create a bar graph
plt.figure(figsize=(10, 6))
plt.bar(cluster_sizes.index, cluster_sizes.values, color=colors, width=0.5, alpha=0.7)
plt.xlabel('Cluster', fontsize=12)
plt.ylabel('Number of Users', fontsize=12)
plt.title('Number of Users in Each Cluster', fontsize=14)
plt.xticks(cluster_sizes.index, cluster_names)  # Assigning cluster names to x-axis ticks
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

D. Task 4

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

dfcustomerengagement = pd.read_csv(r'topic4_edge_list_10000_customers.csv')

dftrendingenres = pd.read_csv(r'topic4_node_list_genre_10000_customers.csv')

dfanalysis= pd.merge(dfcustomerengagement, dftrendingenres, left_on='Target', right_on='Label', how='left')

dfanalysis['Date'] = pd.to_datetime(dfanalysis['Date'])
dfanalysis

dfanalysis['Year'] = dfanalysis['Date'].dt.year
dfanalysis['Month'] = dfanalysis['Date'].dt.month
peakcustomerengagement = dfanalysis.groupby(['Year', 'Month']).size().reset_index(name='Count').sort_values(by='Count',
ascending=False)
print("Peak Interaction Periods:")
peakcustomerengagement.to_csv('peakcustomerengagement.csv')
peakcustomerengagement

year_month_counts = peakcustomerengagement
year_month_counts['Date'] = pd.to_datetime(year_month_counts[['Year', 'Month']].assign(day=1))

year_month_counts.set_index('Date', inplace=True)

year_month_counts.drop(['Year', 'Month'], axis=1, inplace=True)

plt.figure(figsize=(10, 6))
year_month_counts.plot()
plt.title('Temporal Time Series Plot')
plt.xlabel('Date')
plt.ylabel('Count')
```

```
plt.grid(True)
peak_value = year_month_counts['Count'].max()

plt.axhline(y=peak_value, color='r', linestyle='--', label=f'Maximum: {peak_value}')
plt.legend()
plt.savefig('Peak interaction.png')
plt.show()

df1 = peakcustomerengagement

dfcustomerengagement = pd.read_csv(r'topic4_edge_list_10000_customers.csv')
dftrendingenres = pd.read_csv(r'topic4_node_list_genre_10000_customers.csv')
df2= pd.merge(dfcustomerengagement, dftrendingenres, left_on='Target', right_on='Label', how='left')

df2['Date'] = pd.to_datetime(df2['Date'])
df2['Month'] = df2['Date'].dt.month
df2['Year'] = df2['Date'].dt.year

merged_df = pd.merge(df1, df2, on='Date')

genres_stacked = merged_df['MovieGenre'].str.split('|', expand=True).stack().reset_index(level=1, drop=True)

merged_df = merged_df.drop('MovieGenre', axis=1).join(genres_stacked.rename('Genre'))

genre_counts = merged_df.groupby(['Year', 'Month', 'Genre']).size().reset_index(name='Count').sort_values(by='Count',
ascending=False)
genre_counts.to_csv('genre_counts.csv')
genre_counts.head(10)
```