

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THÔNG THÔNG TIN



ĐÒ ÁN MÔN HỌC
MẠNG XÃ HỘI

ĐỀ TÀI
HỆ THỐNG ĐỀ XUẤT PHIM
DỰA TRÊN THUẬT TOÁN MẠNG XÃ HỘI

Giảng viên: ThS. Thái Bảo Trân

Lớp: IS353.P12.HTCL

Thành viên: Nhóm 1

07 – 21520596 – Trần Thị Kim Anh

10 – 21521049 – Hồ Quang Lâm

12 – 21521586 – Lê Thị Lệ Trúc

16 – 21521882 – Lê Minh Chánh

Thành phố Hồ Chí Minh, tháng 12 năm 2024

LỜI CẢM ƠN

Đầu tiên, nhóm chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới giảng viên Thái Bảo Trân - người đã giảng dạy và chia sẻ rất nhiều kiến thức cũng như các ví dụ thực tiễn trong các bài giảng. Cô đã hướng dẫn cho chúng em làm bài tập, sửa chữa và đóng góp nhiều ý kiến quý báu giúp chúng em seminar của mình.

Bộ môn Mạng xã hội là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Tuy nhiên, do vốn kiến thức chuyên môn còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, chúng em rất mong nhận được sự góp ý, chỉ bảo thêm của Cô nhằm hoàn thiện những kiến thức của mình để nhóm chúng em có thể dùng làm hành trang thực hiện tiếp các đề tài khác trong tương lai cũng như là trong học tập và làm việc sau này.

Một lần nữa, nhóm xin gửi đến cô, bạn bè lời cảm ơn đặc biệt chân thành và tốt đẹp nhất!

Thành phố Hồ Chí Minh, tháng 12 năm 2024

Nhóm sinh viên thực hiện

Trần Thị Kim Anh

Hồ Quang Lâm

Lê Thị Lệ Trúc

Lê Minh Chánh

NHẬN XÉT CỦA GIẢNG VIÊN

MỤC LỤC

LỜI CẢM ƠN	2
NHẬN XÉT CỦA GIẢNG VIÊN	3
DANH MỤC HÌNH ẢNH VÀ BẢNG	6
Phần 1. MÔ TẢ BỘ DỮ LIỆU	8
1.1. Nguồn bộ dữ liệu	8
1.2. Bảng mô tả thuộc tính.....	8
1.3. Xác định bài toán	9
1.4. Tổng quan dữ liệu	9
Phần 2. CÁC ĐỘ THỊ CƠ BẢN	10
2.1. Độ thị hai phía	10
2.2. Độ thị một phía (đơn đồ thị vô hướng).....	12
Phần 3. CÁC THUẬT TOÁN TRUNG TÂM	15
3.1. Degree centrality	15
3.2. Betweenness centrality	16
3.3. Closeness centrality	18
3.4. Harmonic Centrality	20
3.5. Eigenvector	21
3.6. Pagerank	22
3.7. Thực hiện các độ đo trên dữ liệu	24
3.7.1. Tính toán các độ đo trên đồ thị	24
3.7.2. Tìm key player trong mạng xã hội.....	25
Phần 4. THUẬT TOÁN PHÂN CỤM CỘNG ĐỒNG.....	28
4.1. Hàm đánh giá thuật toán.....	28

4.1.1.	Hàm Modularity	28
4.2.	Girvan Newman.....	28
4.2.1.	Lý thuyết.....	28
4.2.2.	Thực hiện trên dữ liệu.....	32
4.3.	Louvain.....	36
4.3.1.	Lý thuyết.....	36
4.3.2.	Thực hiện trên dữ liệu.....	36
Phần 5.	DỰ ĐOÁN LIÊN KẾT	39
5.1.	Common Neighbors (CN)	39
5.2.	Jaccard Coefficient (JC)	39
5.3.	Adamic-Adar Index (AA)	40
5.4.	Preferential Attachment (PA)	41
5.5.	Nhận xét	41
5.6.	Thực hiện trên dữ liệu	42
Phần 6.	MÔ HÌNH LAN TRUYỀN THÔNG TIN	45
6.1.	Mô hình IC	45
6.2.	Thực hiện trên dữ liệu	46
Phần 7.	WEBSITE	50
7.1.	Công nghệ sử dụng.....	50
7.2.	Cài đặt API	50
7.3.	Giao diện Website	51
	PHÂN CÔNG CÔNG VIỆC	52
	TÀI LIỆU THAM KHẢO	53

DANH MỤC HÌNH ẢNH VÀ BẢNG

Hình 1.1: Đọc dữ liệu và bắt đầu tạo đồ thị.....	9
Hình 2.1. Minh họa đồ thị hai phía dạng không đầy đủ và đầy đủ	10
Hình 2.2. Trực quan đồ thị hai phía.....	11
Hình 2.3: Kết quả trực quan đồ thị hai phía (500 dòng)	11
Hình 2.4: Minh họa đồ thị một phía (đơn đồ thị vô hướng).....	12
Hình 2.5: Dùng phép chiếu để tạo người đồ thị các user	12
Hình 2.6: Trực quan đơn đồ thị vô hướng	12
Hình 2.7: Kết quả trực quan đơn đồ thị vô hướng (mạng lưới user).....	13
Hình 2.8: Xem số đỉnh và số cạnh của đồ thị	13
Hình 2.9: Xem bậc cao nhất và đỉnh có bậc cao nhất.....	14
Hình 3.1: Đồ thị minh họa cho các độ đo trung tâm (Degree Centrality)	15
Hình 3.2: Đồ thị minh họa cho các độ đo trung tâm (Betweenness Centrality)	17
Hình 3.3: Đồ thị minh họa cho các độ đo trung tâm (Closeness Centrality).....	19
Hình 3.4: Đồ thị minh họa cho các độ đo trung tâm (Harmonic Centrality)	21
Hình 3.5: Đồ thị minh họa cho các độ đo trung tâm (PageRank)	23
Hình 3.6: Đoạn mã tính các độ đo trung tâm bằng Python	24
Hình 3.7: Kết quả tính độ đo trung tâm tất cả các node	24
Hình 3.8: Hàm tìm đỉnh có các độ đo trung tâm cao nhất (key player)	25
Hình 3.9: Đoạn mã trực quan đồ thị với key player là đỉnh nổi bật	26
Hình 3.10: Đoạn mã thực thi tìm keyplayer	26
Hình 3.11: Kết quả keyplayer từ các kết quả độ đo	26
Hình 3.12: Kết quả trực quan đồ thị với đỉnh key player.....	27
Hình 4.1: Ví dụ về các cộng đồng trong mạng xã hội	29
Hình 4.2: Đoạn mã tìm giá trị k với thuật toán Girvan Newman	32
Hình 4.3: Kết quả modularity theo các giá trị k	33
Hình 4.4: Đoạn mã vẽ biểu đồ trực quan kết quả phân chia cộng đồng	33
Hình 4.5: Kết quả trực quan phân chia cộng đồng theo từng giai đoạn k chính....	34
Hình 4.6: Đoạn mã gợi ý phim cho người dùng từ kết quả khám phá cộng đồng ...	35

Hình 4.7: Đoạn mã khám phá cộng đồng bằng thuật toán Louvain	36
Hình 4.8: Trực quan kết quả phân chia cộng đồng.....	37
Hình 4.9: Đoạn mã gợi ý phim cho người dùng từ kết quả cộng đồng	38
Hình 5.1: Đoạn mã thực hiện dự đoán liên kết và gợi ý phim từ kết quả	43
Hình 5.2: Kết quả ứng dụng dự đoán liên kết vào gợi ý phim cho người dùng	44
Hình 6.1: Đoạn mã hiển thị mô hình lan truyền thông tin IC	46
Hình 6.2: Trực quan kết quả các đỉnh bị ảnh hưởng bởi lan truyền	47
Hình 6.3: Đoạn mã gợi ý phim từ các đỉnh bị ảnh hưởng bởi lan truyền	48
Hình 6.4: Kết quả gợi ý phim từ quá trình lan truyền và gợi ý	49
Hình 7.1: Giao diện API được triển khai trên host	50
Hình 7.2: Giao diện tương tác trang web được triển khai trên host.....	51
Hình 7.3: Chọn thuật toán, mô hình để gợi ý	51
Bảng 1: Mô tả chi tiết các thuộc tính bộ dữ liệu	8
Bảng 2: Bảng kết quả ví dụ minh họa về Degree Centrality.....	16
Bảng 3: Bảng tính thủ công ví dụ về Betweenness Centrality.....	17
Bảng 4: Bảng tính thủ công ví dụ về Closeness Centrality	20
Bảng 5: Bảng tính thủ công ví dụ về Harmonic Centrality	21
Bảng 6: Bảng tính thủ công ví dụ về PageRank	23
Bảng 7: Bảng kết quả tính Edge Betweenness	32

Phần 1. MÔ TẢ BỘ DỮ LIỆU

1.1. Nguồn bộ dữ liệu

Bộ dữ liệu được công bố trên website của GroupLens - một phòng nghiên cứu tại Đại học Minnesota, chuyên về các lĩnh vực như tính toán xã hội, hệ thống gợi ý và cộng đồng trực tuyến. Bộ dữ liệu có tên là [MovieLens 32M](#), có tổng cộng hơn 32 triệu lượt đánh giá phim. Tập dữ liệu chuẩn ổn định với 87.585 phim bởi 200.948 người dùng. Thời gian được thu thập 10/2023, thời gian công bố: 05/2024. Bộ dataset MovieLens có nhiều file nhưng nhóm chỉ sử dụng 3 file csv gồm:

- links.csv: Gồm các thuộc tính movieId, imdbId, tmdbId
- movies.csv: Gồm các thuộc tính movieId, title, genres
- ratings.csv: Gồm các thuộc tính userId, movieId, rating, timestamp

Sau đó nhóm thực hiện gom 3 file dữ liệu lại thành 1 file thống nhất với 32.000.204 dòng dữ liệu × 7 cột thuộc tính.

Vì đây là một bộ dữ liệu khá lớn khó cho việc thực hiện nên nhóm quyết định lọc những dữ liệu gần đây để triển khai (năm ra mắt phim từ 2022 và thời gian đánh giá phim từ 02/09/2023).

Sau đó từ thuộc tính tmdbId, sử dụng [TMDB API](#) để thu thập thêm dữ liệu từ các bộ phim để phục vụ cho quá trình xây dựng website gợi ý phim.

1.2. Bảng mô tả thuộc tính

Bảng 1: Mô tả chi tiết các thuộc tính bộ dữ liệu

Mô tả thuộc tính (5345 dòng × 7 cột)		
STT	Thuộc tính	Mô tả
1	userId	Mã người dùng đã đánh giá phim
2	rating	Số điểm đánh giá (thang 5)
3	timestamp	Thời gian đánh giá
4	tmdbId	Mã phim theo TMDB (The Movie Database)
5	title	Tên phim

6	poster	Đường dẫn ảnh poster của phim
7	date_published	Ngày phim ra mắt

1.3. Xác định bài toán

- **Input:** Bộ dữ liệu được công bố trên website của GroupLens với 2 thuộc tính chính userId và tmdbId.
- **Output:** Đưa ra độ đo, đưa ra cộng đồng phục vụ cho việc phân tích mạng xã hội của dataset, thực hiện dự đoán liên kết và thực hiện mô hình lan truyền thông tin trong mạng xã hội, từ đó gợi ý phim cho người dùng.

1.4. Tổng quan dữ liệu

```

● ● ●

1 n = 500
2
3 ratings = pd.read_csv('Dataset.csv', sep=',', header=0)[:n]
4 ratings = ratings.dropna()
5
6 # Lấy danh sách người dùng và phim
7 users = ratings['userId'].unique()
8 movies = ratings['tmdbId'].unique()
9 print('Number of users:', len(users))
10 print('Number of movies:', len(movies))
11 print('Number of ratings:', len(ratings))
12
13 # Tạo danh sách cạnh
14 edges = list(zip(ratings['userId'], ratings['tmdbId']))
15
16 # Tạo đồ thị hai phía (users và movies)
17 G = nx.Graph()
18
19 G.add_nodes_from(users, bipartite=0) # Nhóm người dùng
20 G.add_nodes_from(movies, bipartite=1) # Nhóm phim
21
22 G.add_edges_from(edges)

```

Hình 1.1: Đọc dữ liệu và bắt đầu tạo đồ thị

⇒ Kết quả:

Number of users: 161

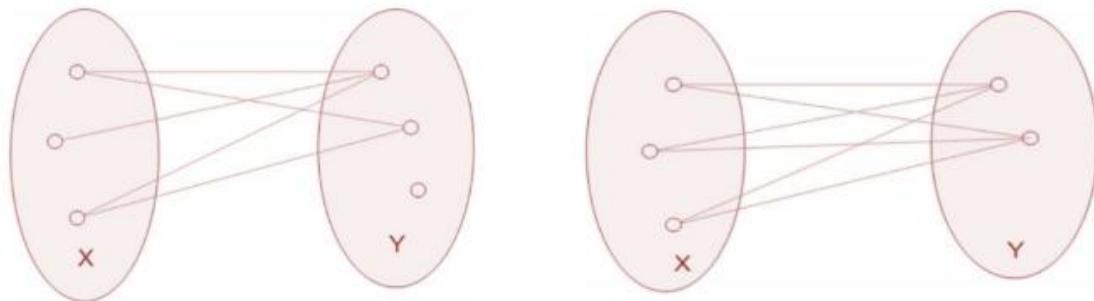
Number of movies: 184

Number of ratings: 500

Phần 2. CÁC ĐỒ THỊ CƠ BẢN

2.1. Đồ thị hai phía

- Đồ thị: $G = (V, E)$ gọi là đồ thị hai phía
 - $V = X \cup Y$, $X \neq \emptyset$, $Y \neq \emptyset$, $X \cap Y = \emptyset$
 - Mỗi cạnh của G sẽ có một đỉnh thuộc X và một đỉnh thuộc Y
- Đồ thị hai phía được gọi là đầy đủ nếu và chỉ nếu mỗi đỉnh thuộc X sẽ nối với mỗi đỉnh thuộc Y
- $|X| = m$, $|Y| = n$
- Ký hiệu là: $K_{m,n}$
- Số cạnh là $m*n$

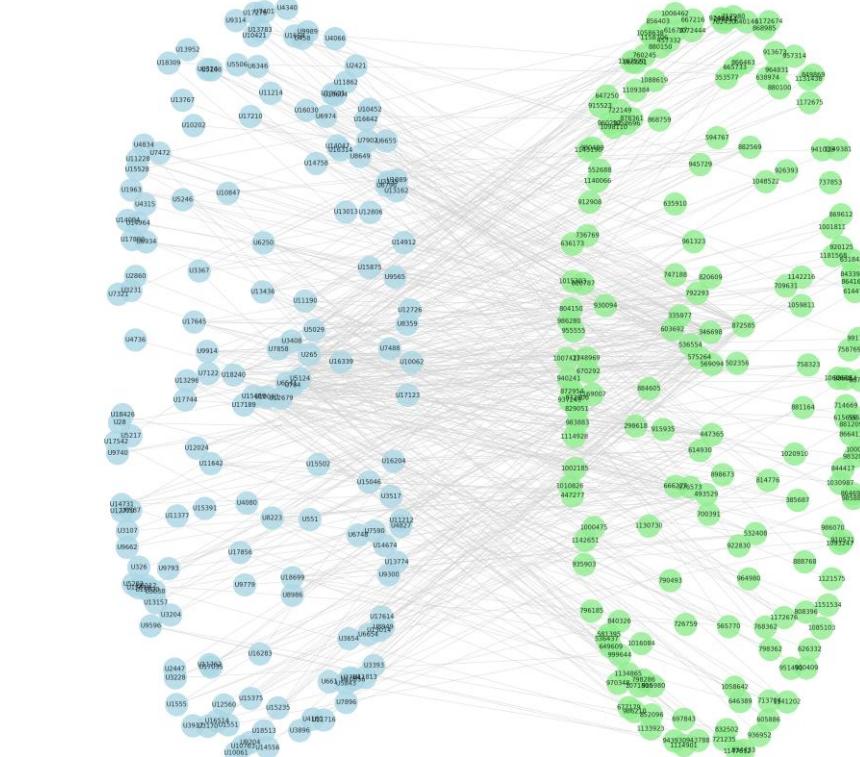


Hình 2.1. Minh họa đồ thị hai phía dạng không đầy đủ và đầy đủ

- Vẽ đồ thị 2 phía:
 - Node: User và Phim
 - Edge: Người dùng đánh giá bộ phim

```
1 plt.figure(figsize=(30, 30))
2
3 # Sử dụng bố cục bipartite_layout
4 pos = nx.spring_layout(B, k=0.7)
5
6 # Tăng khoảng cách giữa các nhóm
7 for key, value in pos.items():
8     if key in users:
9         pos[key][0] -= 1.5 # Di chuyển nhóm user sang trái nhiều hơn
10    else:
11        pos[key][0] += 1.5 # Di chuyển nhóm movie sang phải nhiều hơn
12
13 nx.draw(B, pos,
14           with_labels=True,
15           node_color=['lightblue' if n in users else 'lightgreen' for n in B.nodes()],
16           node_size=3000, # Tăng kích thước nút để dễ nhận diện hơn
17           font_size=15, # Tăng kích thước phông chữ
18           edge_color='lightgray', # Đổi màu cạnh để dễ nhìn hơn
19           alpha=0.8, # Tăng độ trong suốt của cạnh một chút để giảm sự rối mắt
20           width=1.2) # Giảm độ dày cạnh để tránh quá đậm
21
22 plt.show()
```

Hình 2.2. Trực quan đồ thị hai phía

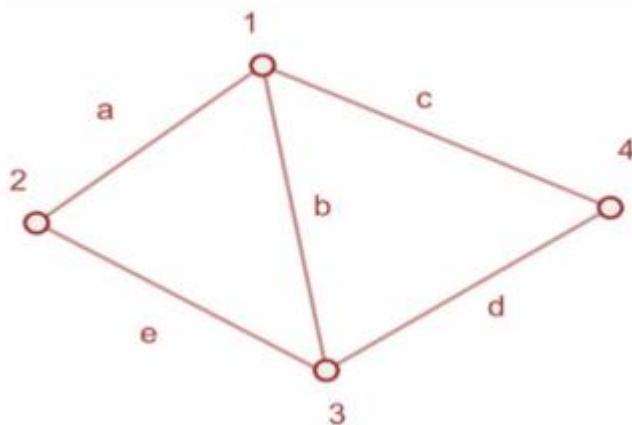


Hình 2.3: Kết quả trực quan đồ thị hai phía (500 dòng)

⇒ Khi nhìn vào đồ thị 2 phía, ta có thể thấy một bộ phim có thể được đánh giá bởi nhiều người dùng và người dùng cũng có thể đánh giá nhiều bộ phim.

2.2. Đồ thị một phía (đơn đồ thị vô hướng)

- Đồ thị G là một tập hợp gồm các đỉnh và các cạnh
- Ký hiệu $G = (V, E)$, trong đó:
 - V : là tập các đỉnh của đồ thị
 - E : là tập hợp các cạnh của đồ thị



Hình 2.4: Minh họa đồ thị một phía (đơn đồ thị vô hướng)

• Vẽ đồ thị 1 phía:

- Node: User
- Edge: Hai user đánh giá cùng một bộ phim sẽ tạo thành 1 cạnh



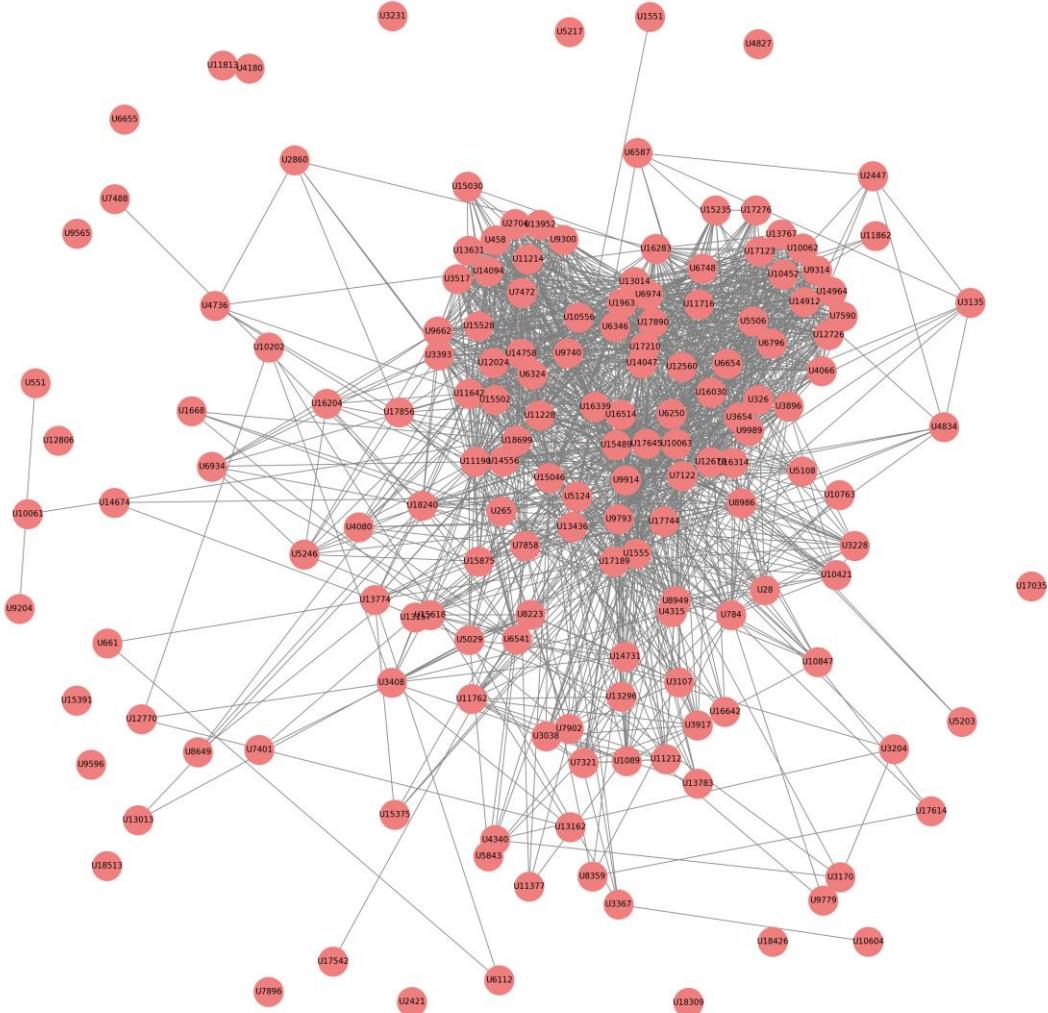
```
1 # Tạo đồ thị chiếu người dùng-phim
2 user_movie = nx.bipartite.weighted_projected_graph(B, users)
```

Hình 2.5: Dùng phép chiếu để tạo người đồ thị các user



```
1 plt.figure(figsize=(30, 30))
2
3 # Vẽ đồ thị one-mode
4 pos = nx.spring_layout(user_movie, k=0.7)
5 nx.draw(user_movie, pos, with_labels=True, node_color='lightcoral', node_size=3000, font_size=15, edge_color='gray', width=1.5)
6 plt.show()
```

Hình 2.6: Trực quan đơn đồ thị vô hướng



Hình 2.7: Kết quả trực quan đơn đồ thị vô hướng (mạng lưới user)

- Một số thuộc tính:



```

1 # Số đỉnh số cạnh
2 print("Số đỉnh của đồ thị:", user_movie.number_of_nodes())
3 print("Số cạnh của đồ thị:", user_movie.number_of_edges())

```

Hình 2.8: Xem số đỉnh và số cạnh của đồ thi

⇒ Kết quả:

Số đỉnh của đồ thị: 161

Số cạnh của đồ thị: 1827



```
1 degree_dict = dict(user_movie.degree())
2
3 max_degree_value = max(degree_dict.values())
4 print(f'Số mối quan hệ nhiều nhất là {max_degree_value}')
5
6 print('Các user có số quan hệ nhiều nhất là: ', end='')
7 print(', '.join(f'{key}' for key, value in degree_dict.items() if value == max_degree_value))
```

Hình 2.9: Xem bậc cao nhất và đỉnh có bậc cao nhất

⇒ Kết quả:

Số mối quan hệ nhiều nhất là 94

Các user có số quan hệ nhiều nhất là: "U15489"

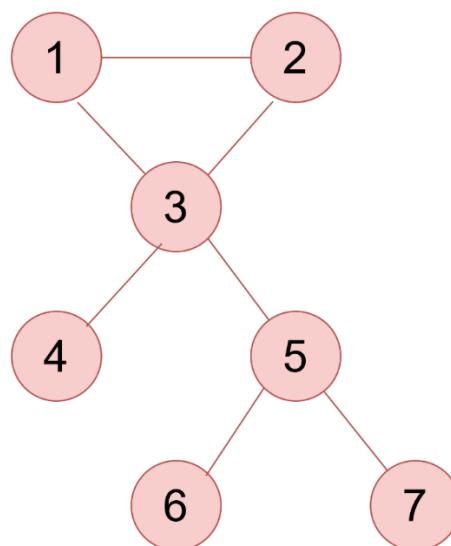
Phần 3. CÁC THUẬT TOÁN TRUNG TÂM

3.1. Degree centrality

- Còn có tên khác là độ đo bậc trong lý thuyết đồ thị, thể hiện số cạnh kết nối trực tiếp với một đỉnh.
- Đỉnh nào có nhiều kết nối hơn => Đỉnh đó bậc cao hơn và tầm ảnh hưởng của nó lớn hơn trong mạng lưới.
- Độ đo trung tâm về bậc ký hiệu $D(v)$ được tính bằng:

$$D(v) = \frac{\deg(v)}{n-1}$$

- **Ứng dụng:**
 - Mạng xã hội: Độ đo degree cho biết một người dùng có bao nhiêu mối liên hệ trực tiếp. Những người có degree cao thường là những người có nhiều kết nối và tầm ảnh hưởng trong cộng đồng.
 - Mạng máy tính: Trong một mạng lưới, các nút có degree cao đóng vai trò quan trọng trong việc kết nối và truyền tải dữ liệu.
 - Phân tích hệ thống: Degree có thể giúp xác định điểm nút quan trọng cần bảo vệ trong các hệ thống phức tạp như hệ thống điện, hệ thống giao thông.
- **Ví dụ:**
 - Ta có đồ thị vô hướng như sau:



Hình 3.1: Đồ thị minh họa cho các độ đo trung tâm (Degree Centrality)

⇒ Ta tính được Degree Centrality của các Node:

Bảng 2: Bảng kết quả ví dụ minh họa về Degree Centrality

Node	Degree Centrality	Normalizes Degree Centrality
1	2	0.3333333333
2	3	0.5
3	4	0.6666666667
4	1	0.1666666667
5	4	0.6666666667
6	1	0.1666666667
7	1	0.1666666667

⇒ NODE 3 VÀ NODE 5 LÀ KEY PLAYER

3.2. Betweenness centrality

- Betweenness Centrality đo lường mức độ mà một đỉnh (node) đóng vai trò trung gian trong mạng lưới.
- Tính toán số lượng đường đi ngắn nhất giữa hai đỉnh khác mà đi qua đỉnh đó.
- Đỉnh có betweenness cao đóng vai trò quan trọng trong việc kết nối các phần khác nhau của mạng, như một "người trung gian" truyền tải thông tin.

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Trong đó:

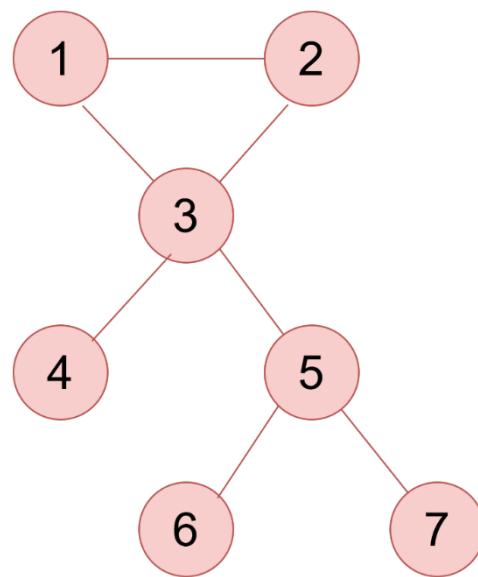
$C_B(v)$: Betweenness của đỉnh v

σ_{st} : Số đường đi ngắn nhất từ s đến t

$\sigma_{st}(v)$: Số đường đi ngắn nhất từ s đến t mà qua v

- Ví dụ:**

- Ta có đồ thị vô hướng như sau:

**Hình 3.2:** Đồ thị minh họa cho các độ đo trung tâm (Betweenness Centrality)

⇒ Ta tính được Betweenness Centrality của các Node:

Bảng 3: Bảng tính thủ công ví dụ về Betweenness Centrality

	1	2	3	4	5	6	7
1-2	-	-	0/1	0/1	0/1	0/1	0/1
1-3	-	0/1	-	0/1	0/1	0/1	0/1
1-4	-	0/1	1/1	-	0/1	0/1	0/1
1-5	-	0/1	1/1	0/1	-	0/1	0/1
1-6	-	0/1	1/1	0/1	1/1	-	0/1
1-7	-	0/1	1/1	0/1	1/1	0/1	-
2-3	0/1	-	-	0/1	0/1	0/1	0/1
2-4	0/1	-	1/1	-	0/1	0/1	0/1
2-5	0/1	-	1/1	0/1	-	0/1	0/1
2-6	0/1	-	1/1	0/1	1/1	-	0/1
2-7	0/1	-	1/1	0/1	1/1	0/1	-

3-4	0/1	0/1	-	-	0/1	0/1	0/1
3-5	0/1	0/1	-	0/1	-	0/1	0/1
3-6	0/1	0/1	-	0/1	1/1	-	0/1
3-7	0/1	0/1	-	0/1	1/1	0/1	-
4-5	0/1	0/1	1/1	-	-	0/1	0/1
4-6	0/1	0/1	1/1	-	1/1	-	0/1
4-7	0/1	0/1	1/1	-	1/1	0/1	-
5-6	0/1	0/1	0/1	0/1	0/1	0/1	0/1
5-7	0/1	0/1	0/1	0/1	0/1	0/1	0/1
6-7	0/1	0/1	0/1	0/1	1/1	0/1	0/1
	0	0	11	0	9	0	0

⇒ NODE 3 LÀ KEY PLAYER

3.3. Closeness centrality

- Độ đo Closeness Centrality cho biết mức độ gần gũi của một đỉnh (node) với tất cả các đỉnh còn lại trong đồ thị.
- Nó được tính dựa trên tổng khoảng cách ngắn nhất từ đỉnh đó đến tất cả các đỉnh khác.
- Closeness cao nghĩa là đỉnh có thể tiếp cận các đỉnh khác nhanh chóng trong mạng lưới.

$$C_C(v) = \frac{n-1}{\sum_{u \neq v} d(v,u)}$$

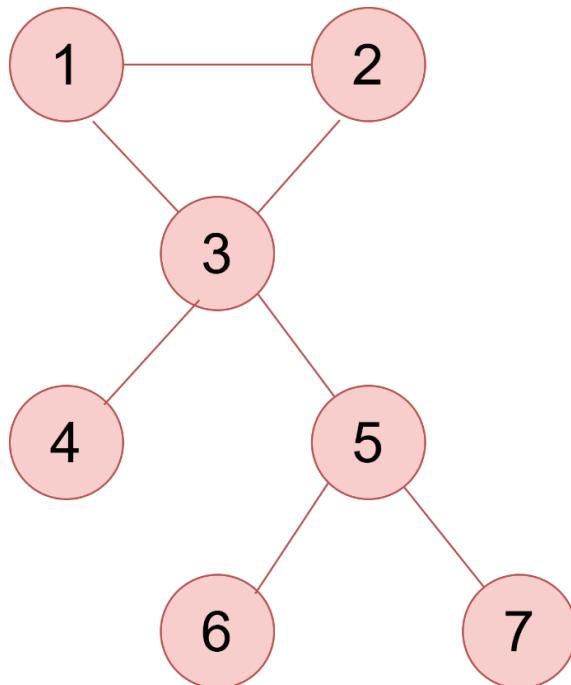
Trong đó:

$C_C(v)$: Closeness của đỉnh v

n: số đỉnh của đồ thị

d (v, u): Khoảng cách ngắn nhất từ đỉnh v đến đỉnh u

- **Ứng dụng:**
 - Closeness Centrality được dùng để xác định những cá nhân có thể kết nối và tiếp cận các thành viên khác trong mạng lưới một cách nhanh chóng. Những cá nhân này thường là những người có khả năng truyền tải thông tin một cách nhanh chóng và rộng rãi trong cộng đồng.
 - Trong các hệ thống phân phối hàng hóa, closeness được sử dụng để xác định những vị trí trung tâm có khả năng cung cấp hàng hóa nhanh chóng và hiệu quả tới nhiều khu vực khác. Ví dụ, các kho hàng có closeness cao sẽ giúp giảm thời gian giao hàng.
 - Trong nghiên cứu phân tích dữ liệu về sinh học, closeness có thể được sử dụng để xác định những gene hoặc protein quan trọng trong các mạng lưới sinh học, có ảnh hưởng lớn đến các phần khác của mạng lưới.
- **Ví dụ:** Ta có đồ thị vô hướng như sau:



Hình 3.3: Đồ thị minh họa cho các độ đo trung tâm (Closeness Centrality)

⇒ Ta tính được Closeness Centrality của các Node:

Bảng 4: Bảng tính thủ công ví dụ về Closeness Centrality

	1	2	3	4	5	6	7	
1	0	1	1	2	2	3	3	0.5
2	1	0	1	2	2	3	3	0.5
3	1	1	0	1	1	2	2	0.75
4	2	2	1	0	2	3	3	0.4615384615
5	2	2	1	2	0	1	1	0.6666666667
6	3	3	2	3	1	0	2	0.4285714286
7	3	3	2	3	1	2	0	0.4285714286

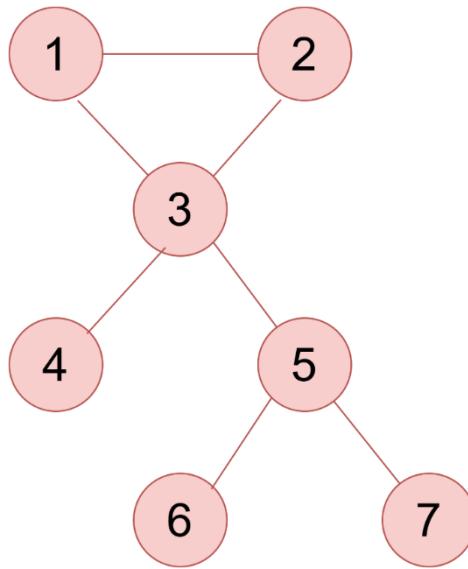
⇒ NODE 3 LÀ KEY PLAYER

3.4. Harmonic Centrality

- Harmonic Centrality là một độ đo trong lý thuyết đồ thị, đánh giá mức độ "gần gũi" của một đỉnh với tất cả các đỉnh khác trong mạng.
- Nó dựa trên khoảng cách nghịch đảo giữa một đỉnh và các đỉnh còn lại, nghĩa là, nếu một đỉnh càng gần với các đỉnh khác thì giá trị Harmonic Centrality của nó càng cao.

$$HC(v) = \sum_{u \neq v} \frac{1}{d(u,v)}$$

- Đồ thị không kết nối dẫn đến closeness là vô hạn dẫn đến giá trị bằng không. Harmonic giải quyết cách vô hạn, đánh giá độ gần gũi bằng kết nối giữa chúng.
- **Ví dụ:** Ta có đồ thị vô hướng như sau:



Hình 3.4: Đồ thị minh họa cho các độ đo trung tâm (Harmonic Centrality)

⇒ Ta tính được Harmonic Centrality của Node X:

Bảng 5: Bảng tính thủ công ví dụ về Harmonic Centrality

	1	2	3	4	5	6	7	HARMONIC
1	0	1	1	2	2	3	3	3.666666667
2	1	0	1	2	2	3	3	3.666666667
3	1	1	0	1	1	2	2	5
4	2	2	1	0	2	3	3	3.166666667
5	2	2	1	2	0	1	1	4.5
6	3	3	2	3	1	0	2	3
7	3	3	2	3	1	2	0	3

⇒ NODE 3 LÀ KEY PLAYER

3.5. Eigenvector

- Eigenvector (vector riêng) của một ma trận là một vector không bằng 0, mà khi được nhân với ma trận đó, nó sẽ chỉ thay đổi độ lớn (và có thể là chiều

hướng) nhưng không thay đổi hướng. Cụ thể, nếu A là một ma trận vuông và v là một vector, thì:

$$Av=\lambda v$$

Trong đó:

A là ma trận.

v là eigenvector.

λ là eigenvalue (giá trị riêng) tương ứng với eigenvector đó.

3.6. Pagerank

- Thuật toán PageRank do Google phát triển để xếp hạng trang web dựa trên cấu trúc liên kết của một trang bằng cách xem xét số lượng và chất lượng các liên kết đến trang đó. Trang có nhiều liên kết chất lượng cao sẽ có thứ hạng cao.
- Công thức thuật toán PageRank:

$$PR(u) = (1 - d) + d \left(\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + \dots + \frac{PR(T_n)}{C(T_n)} \right)$$

Trong đó:

PR(u): PageRank của node u

d: một ngưỡng đặt ra của thuật toán PageRank (thường lấy 0.85)

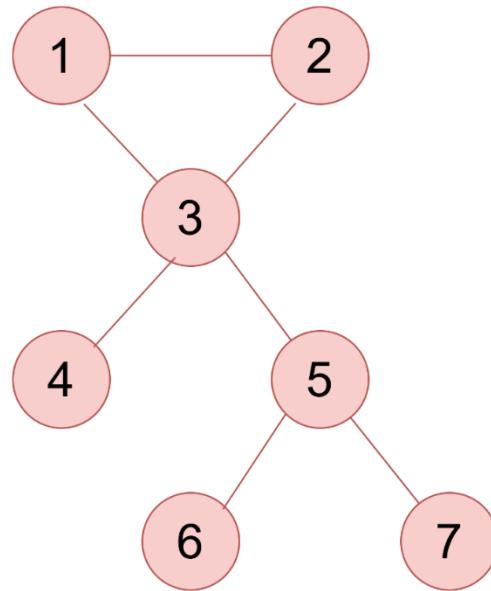
C(Ti): số bậc ra của node Ti

⇒ Dừng khi hội tụ hoặc đạt số lần lặp nhất định

- Khi nào sử dụng thuật toán PageRank?

- *Để xuất tài khoản trên mạng xã hội:* Twitter sử dụng PageRank cá nhân hóa để giới thiệu các tài khoản mà người dùng có thể muốn theo dõi, dựa trên sở thích chung và kết nối.
- *Dự đoán dòng lưu thông và di chuyển của con người:* PageRank được sử dụng để phân tích các giao lộ đường phố, giúp dự đoán hành vi của con người như đỗ xe hoặc kết thúc hành trình.

- *Phát hiện gian lận và bất thường:* Trong ngành chăm sóc sức khỏe và bảo hiểm, PageRank giúp phát hiện các bác sĩ hoặc nhà cung cấp có hành vi bất thường và đưa vào thuật toán học máy
- **Ví dụ:** Ta có đồ thị vô hướng như sau:



Hình 3.5: Đồ thị minh họa cho các độ đo trung tâm (PageRank)

⇒ Ta tính được PageRank của đồ thị như sau:

Bảng 6: Bảng tính thủ công ví dụ về PageRank

INTERATION	1	2	3	4	5	6	7
0	1/7	1/7	1/7	1/7	1/7	1/7	1/7
1	0.107	0.107	0.333	0.035	0.321	0.047	0.047
2	0.136	0.136	0.25	0.083	0.178	0.107	0.107
Sort	3	4	1	7	2	5	6

⇒ NODE 3 LÀ KEY PLAYER

3.7. Thực hiện các độ đo trên dữ liệu

3.7.1. Tính toán các độ đo trên đồ thị



```

1 # Tính toán Degree Centrality
2 degree_centrality = nx.degree_centrality(user_movie)
3 print("Degree Centrality:", degree_centrality)
4
5 # Tính toán Betweenness Centrality
6 betweenness_centrality = nx.betweenness_centrality(user_movie)
7 print("Betweenness Centrality:", betweenness_centrality)
8
9 # Tính toán Closeness Centrality
10 closeness_centrality = nx.closeness_centrality(user_movie)
11 print("Closeness Centrality:", closeness_centrality)
12
13 # Tính toán Harmonic Centrality
14 harmonic_centrality = nx.harmonic_centrality(user_movie)
15 print("Harmonic Centrality:", harmonic_centrality)
16
17 # Tính toán Eigenvector Centrality
18 eigenvector_centrality = nx.eigenvector_centrality(user_movie)
19 print("Eigenvector Centrality:", eigenvector_centrality)
20
21 # Tính toán PageRank
22 pagerank = nx.pagerank(user_movie)
23 print("PageRank:", pagerank)

```

Hình 3.6: Đoạn mã tính các độ đo trung tâm bằng Python

⇒ Kết quả:

```

Degree Centrality: {'U28': 0.1187500000000001, 'U265': 0.1375, 'U326': 0.1062500000000001, 'U458': 0.2062500000000002, 'U551': 0.00625, 'U661': 0.0125, 'U784': 0.1625, 'U1089': 0.08
Betweenness Centrality: {'U28': 0.004961668893766009, 'U265': 0.0059674743047949405, 'U326': 0.0, 'U458': 0.0, 'U551': 0.0, 'U661': 0.0, 'U784': 0.007507263707907966, 'U1089': 0.0, 'U1
Closeness Centrality: {'U28': 0.4566123188405796, 'U265': 0.4582727272727272, 'U326': 0.44375, 'U458': 0.472003745318352, 'U551': 0.00625, 'U661': 0.29105080831408775, 'U784': 0.465036
Harmonic Centrality: {'U10604': 43.45000000000002, 'U4340': 62.50000000000006, 'U14758': 88.33333333333333, 'U4834': 63.416666666666672, 'U8359': 63.50000000000005, 'U13157': 69.083333
Eigenvector Centrality: {'U28': 0.041973863441135656, 'U265': 0.04896897277475858, 'U326': 0.03893270877199253, 'U458': 0.10160047090029915, 'U551': 5.1686315339731980-19, 'U661': 0.08
PageRank: {'U28': 0.004887115644034907, 'U265': 0.006298561855408703, 'U326': 0.004268214839705707, 'U458': 0.006699471556426064, 'U551': 0.006784260515603799, 'U661': 0.00240171753994

```

Hình 3.7: Kết quả tính độ đo trung tâm tất cả các node

3.7.2. Tìm key player trong mạng xã hội

- Hàm tìm các nút có giá trị cao nhất cho từng loại centrality trong đồ thị:



```

1  def find_highest_centrality(graph):
2
3      # Tìm node có giá trị cao nhất cho từng loại centrality
4      max_degree_node = max(degree_centrality, key=degree_centrality.get)
5      max_betweenness_node = max(betweenness_centrality, key=betweenness_centrality.get)
6      max_closeness_node = max(closeness_centrality, key=closeness_centrality.get)
7      max_harmonic_node = max(harmonic_centrality, key=harmonic_centrality.get)
8      max_eigenvector_node = max(eigenvector_centrality, key=eigenvector_centrality.get)
9      max_pagerank_node = max(pagerank, key=pagerank.get)
10
11     centrality_data = [
12         ("Degree", max_degree_node, degree_centrality),
13         ("Betweenness", max_betweenness_node, betweenness_centrality),
14         ("Closeness", max_closeness_node, closeness_centrality),
15         ("Harmonic", max_harmonic_node, harmonic_centrality),
16         ("Eigenvector", max_eigenvector_node, eigenvector_centrality),
17         ("PageRank", max_pagerank_node, pagerank)
18     ]
19     data = {
20         "Centrality Type": [name for name, node, _ in centrality_data],
21         "Node": [node for _, node, _ in centrality_data],
22         "Value": [centrality[node] for _, node, centrality in centrality_data]
23     }
24     df = pd.DataFrame(data)
25     print("Các node có giá trị cao nhất cho từng loại centrality:")
26     print(df)
27
28     return {
29         "Degree": (max_degree_node, degree_centrality[max_degree_node]),
30         "Closeness": (max_closeness_node, closeness_centrality[max_closeness_node]),
31         "Betweenness": (max_betweenness_node, betweenness_centrality[max_betweenness_node]),
32         "Harmonic": (max_harmonic_node, harmonic_centrality[max_harmonic_node]),
33         "Eigenvector": (max_eigenvector_node, eigenvector_centrality[max_eigenvector_node]),
34         "PageRank": (max_pagerank_node, pagerank[max_pagerank_node])
35     }

```

Hình 3.8: Hàm tìm đỉnh có các độ đo trung tâm cao nhất (key player)

- Hàm vẽ đồ thị và hiển thị các nút với giá trị centrality cao nhất:

```

● ● ●

1 def plot_graph_with_centrality(graph, centrality_results):
2     centrality_colors = {
3         "Degree": "red",
4         "Closeness": "blue",
5         "Betweenness": "yellow",
6         "Harmonic": "purple",
7         "Eigenvector": "orange",
8         "PageRank": "cyan"
9     }
10
11 plt.figure(figsize=(30, 30))
12 plt.axis('off')
13
14 pos = nx.spring_layout(graph, k=0.7)
15
16 nx.draw_networkx_edges(graph, pos, alpha=0.5)
17 nx.draw_networkx_labels(graph, pos)
18
19 for centrality, (node, _) in centrality_results.items():
20     nx.draw_networkx_nodes(graph, pos, nodelist=[node], node_color=centrality_colors[centrality], label=centrality, node_size=1500)
21
22 # Draw remaining nodes (non-highlighted) in gray
23 remaining_nodes = set(graph.nodes) - {node for node, _ in centrality_results.values()}
24 nx.draw_networkx_nodes(graph, pos, nodelist=list(remaining_nodes), node_color="gray", alpha=0.5, node_size=3000)
25
26 plt.show()

```

Hình 3.9: Đoạn mã trực quan đồ thị với key player là đỉnh nổi bật

- Tìm và trực quan hóa các nút trong đồ thị người dùng-phim (user - movie) dựa trên các chỉ số centrality khác nhau

● ● ●

```

1 result = find_highest_centrality(user_movie)
2 plot_graph_with_centrality(user_movie, result)

```

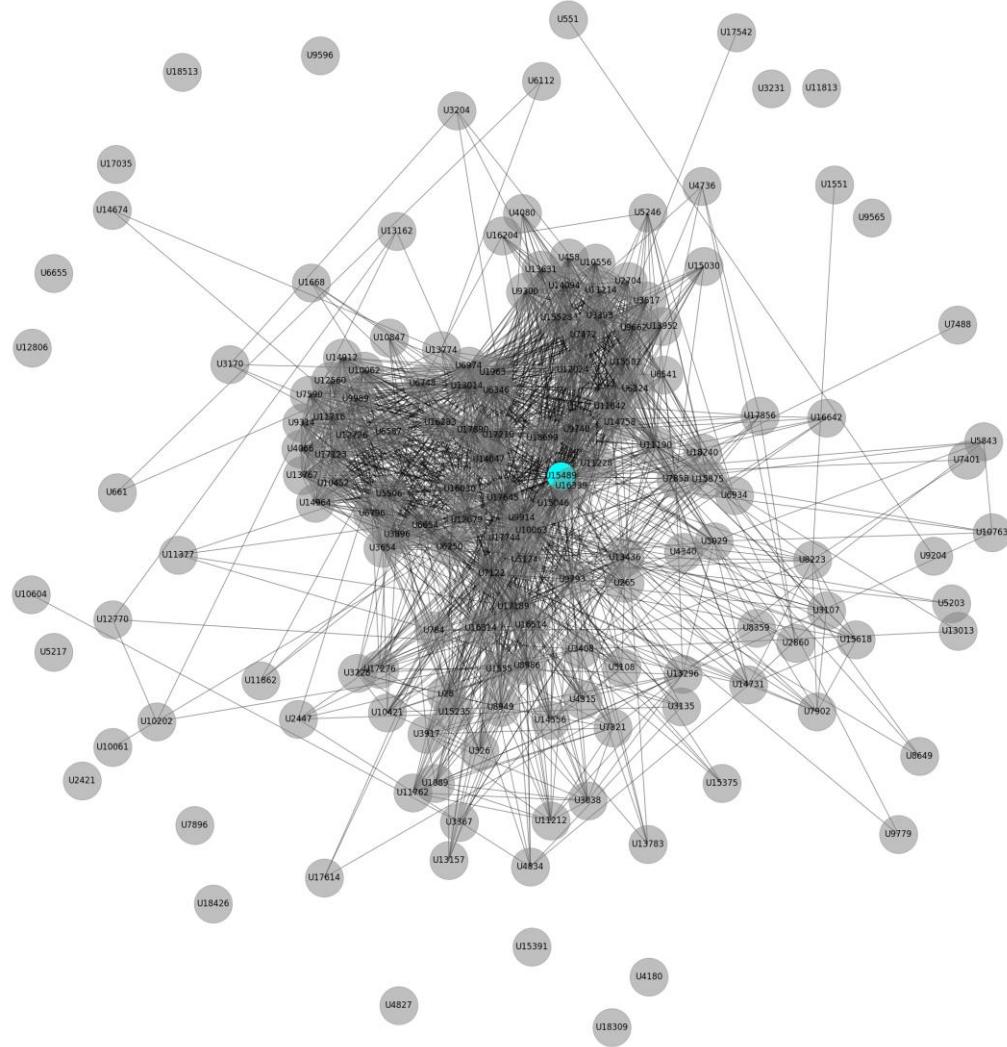
Hình 3.10: Đoạn mã thực thi tìm keyplayer

⇒ Kết quả:

Các node có giá trị cao nhất cho từng loại centrality:

	Centrality Type	Node	Value
0	Degree	U15489	0.587500
1	Betweenness	U15489	0.095502
2	Closeness	U15489	0.639721
3	Harmonic	U15489	116.833333
4	Eigenvector	U15489	0.197542
5	PageRank	U15489	0.029044

Hình 3.11: Kết quả keyplayer từ các kết quả độ đo



Hình 3.12: Kết quả trực quan đồ thị với đỉnh key player

Phần 4. THUẬT TOÁN PHÂN CỤM CỘNG ĐỒNG

4.1. Hàm đánh giá thuật toán

4.1.1. Hàm Modularity

- Modularity là một thước đo thường được sử dụng trong lý thuyết đồ thị để đánh giá chất lượng của việc phân chia các đỉnh thành các cộng đồng.
- Giá trị modularity cao cho thấy rằng có nhiều cạnh hơn trong các cộng đồng so với kỳ vọng, điều này gợi ý rằng cộng đồng có cấu trúc rõ ràng và hợp lý.
- Trong đồ thị vô hướng, các cạnh không có hướng, và modularity được tính dựa trên số lượng cạnh giữa các đỉnh trong cùng cộng đồng. Công thức tính modularity cho đồ thị vô hướng được định nghĩa như sau:

$$Q_u = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(g_i, g_j),$$

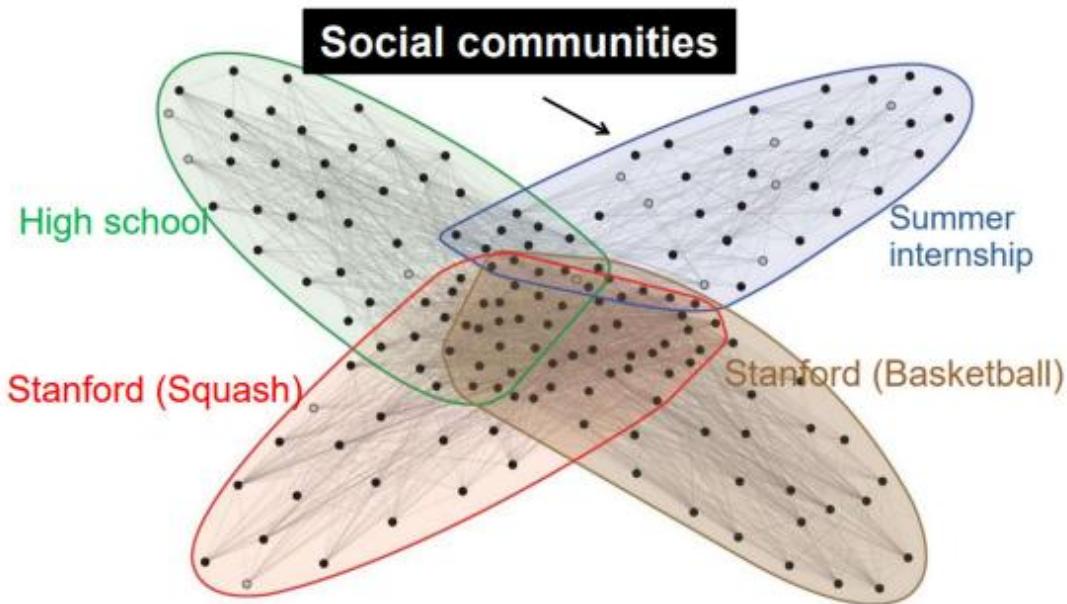
Trong đó:

- A_{ij} là số cạnh giữa đỉnh i và j .
- d_i và d_j lần lượt là bậc của đỉnh i và j .
- m là tổng số cạnh trong đồ thị.
- $\delta(g_i, g_j) = 1$ nếu i và j thuộc một cộng đồng, ngược lại $\delta(g_i, g_j) = 0$.
- $\frac{d_i d_j}{2m}$ là số cạnh trung bình giữa hai đỉnh i và j theo đồ thị cấu hình

4.2. Girvan Newman

4.2.1. Lý thuyết

- Thuật toán Girvan – Newman là một phương pháp phát hiện cộng đồng trong đồ thị, hoạt động bằng cách lặp lại quá trình loại bỏ cạnh có giá trị trung gian lớn nhất, từ đó chia đồ thị thành các cụm cộng đồng nhỏ hơn.

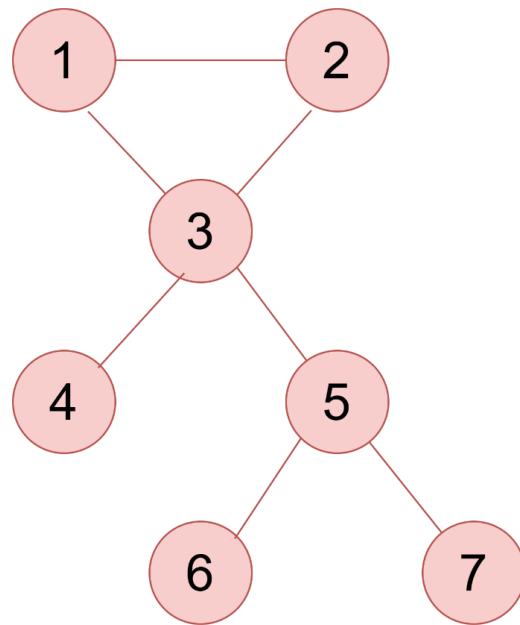


Hình 4.1: Ví dụ về các cộng đồng trong mạng xã hội

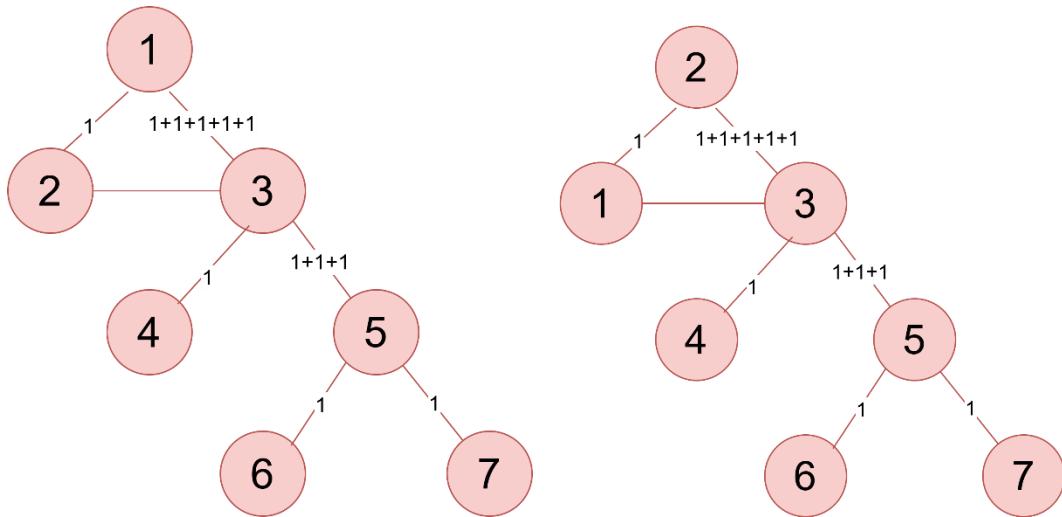
- Các loại cộng đồng:
 - Cộng đồng chồng chéo
 - Cộng đồng tách biệt
- Các bước thực hiện thuật toán Girvan - Newman
 - Tính hệ số trung tâm trung gian của tất cả các cạnh trong mạng.
 - Loại bỏ những cạnh có hệ số cao nhất (Nếu có từ 2 trở lên có hệ số giống nhau ta có thể loại bỏ ngẫu nhiên 1 cạnh hoặc loại bỏ tất cả các cạnh).
 - Tính toán lại hệ số trung tâm trung gian cho tất cả các cạnh bị ảnh hưởng bởi bước 2.
 - Lặp lại bước 2 cho đến khi không còn cạnh nào trong mạng lưới.
- Edge Betweenness: là thước đo đánh giá mức độ quan trọng của một cạnh trong việc giữ cho đồ thị được kết nối – là số lượng đường đi ngắn nhất đi qua cạnh đó.

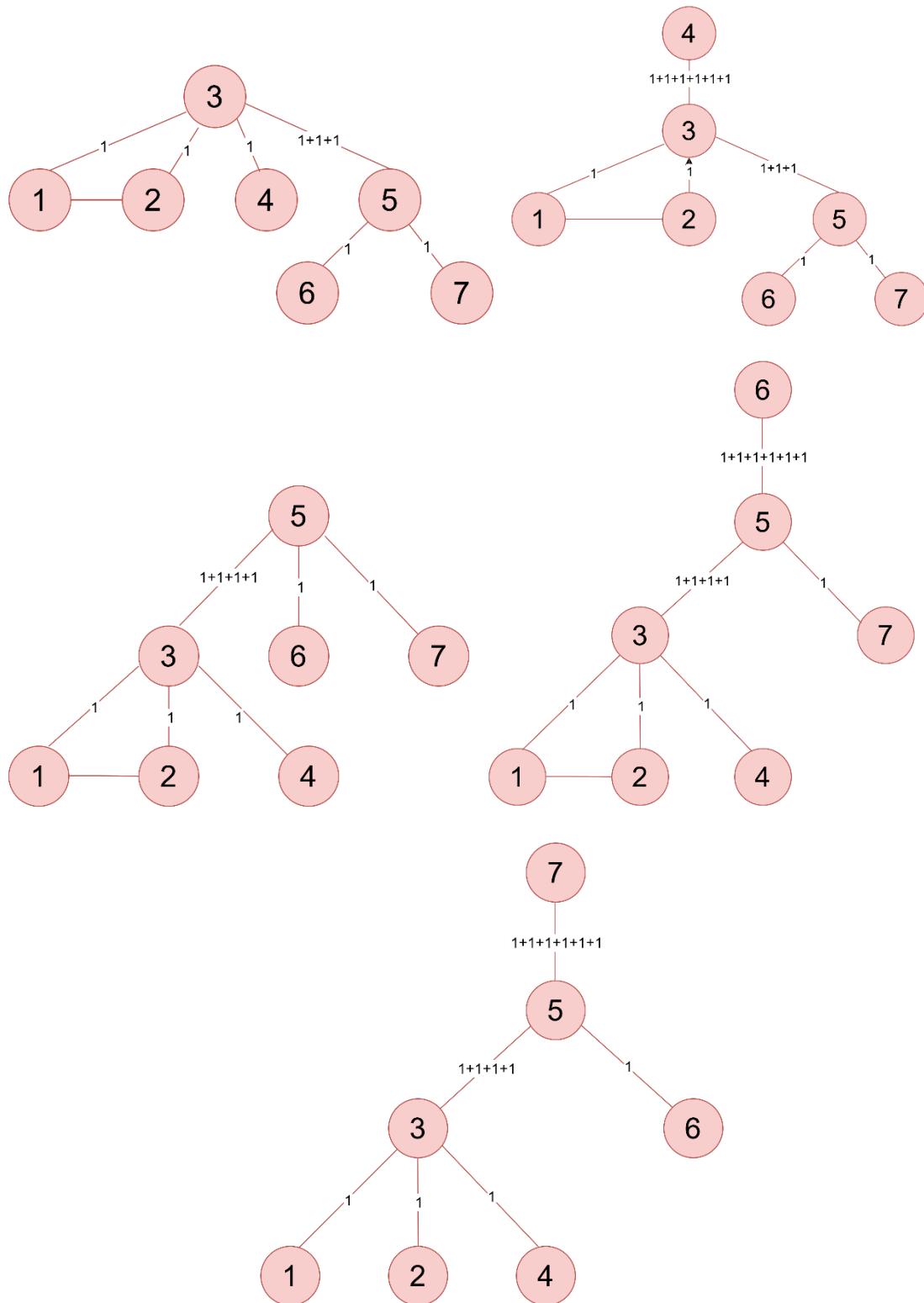
$$B(a, b) = \sum_{x, y \in V} \frac{\text{Số lượng đường đi ngắn nhất giữa } x \text{ và } y \text{ chứa } (a, b)}{\text{Tổng số đường đi ngắn nhất giữa } x \text{ và } y}$$

- **Ví dụ:** Ta có đồ thị vô hướng như sau:



- Ta tính Edge Betweenness





Bảng 7: Bảng kết quả tính Edge Betweenness

Cạnh	Edge Betweenness
1-2	2
1-3	10
2-3	10
3-4	12
3-5	24
5-6	12
5-7	12

4.2.2. Thực hiện trên dữ liệu

- Tìm giá trị K tại bước phân chia cộng đồng với độ liên kết modularity tốt nhất



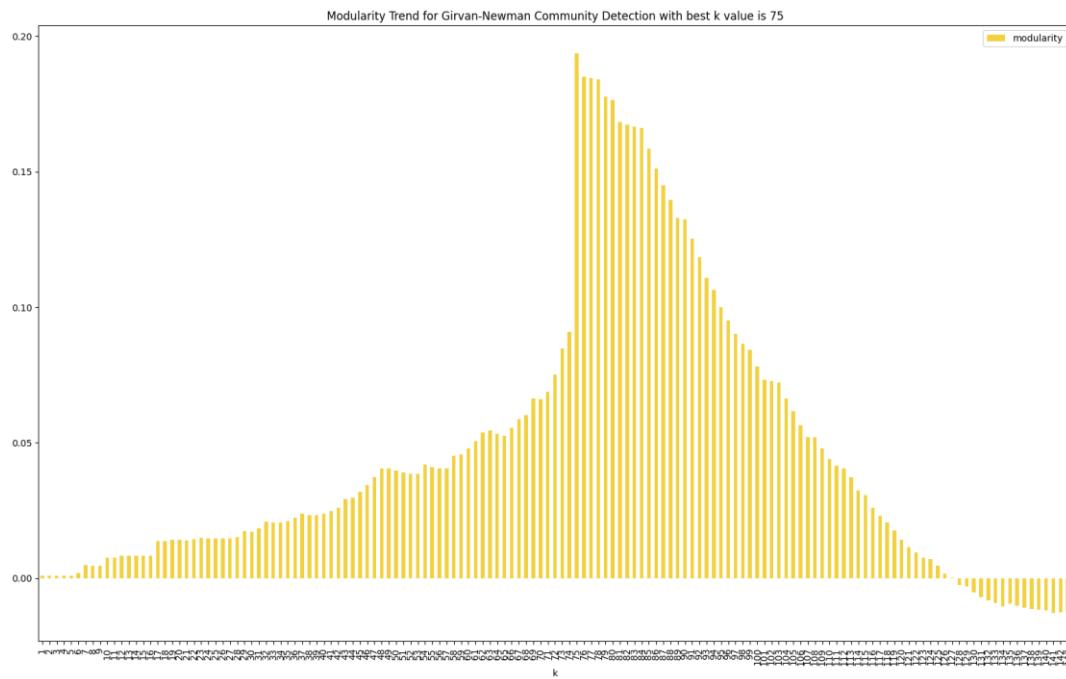
```

1 G = user_movie.copy()
2
3 # Find communities using Girvan-Newman
4 communities = list(nx.community.girvan_newman(G))
5
6 # Calculate modularity for each partition level
7 modularity_df = pd.DataFrame(
8     [
9         [k + 1, nx.community.modularity(G, communities[k])]
10        for k in range(len(community))]
11    ],
12    columns=["k", "modularity"],
13 )
14
15 # Find the k value with the highest modularity
16 best_k = (modularity_df.loc[modularity_df["modularity"]
17                             .idxmax()]["k"]).astype(int)
18
19 fig, ax = plt.subplots(figsize=(20, 12))
20
21 # Plot change in modularity
22 modularity_df.plot.bar(
23     x="k",
24     y="modularity",
25     ax=ax,
26     color="#F2D140",
27     title="Modularity Trend for Girvan-Newman Community Detection with best k value is " + str(best_k),
28 )
29 plt.show()

```

Hình 4.2: Đoạn mã tìm giá trị k với thuật toán Girvan Newman

⇒ Kết quả:



Hình 4.3: Kết quả modularity theo các giá trị k

- Trực quan các cộng đồng được phân chia sau 3 mốc đại diện



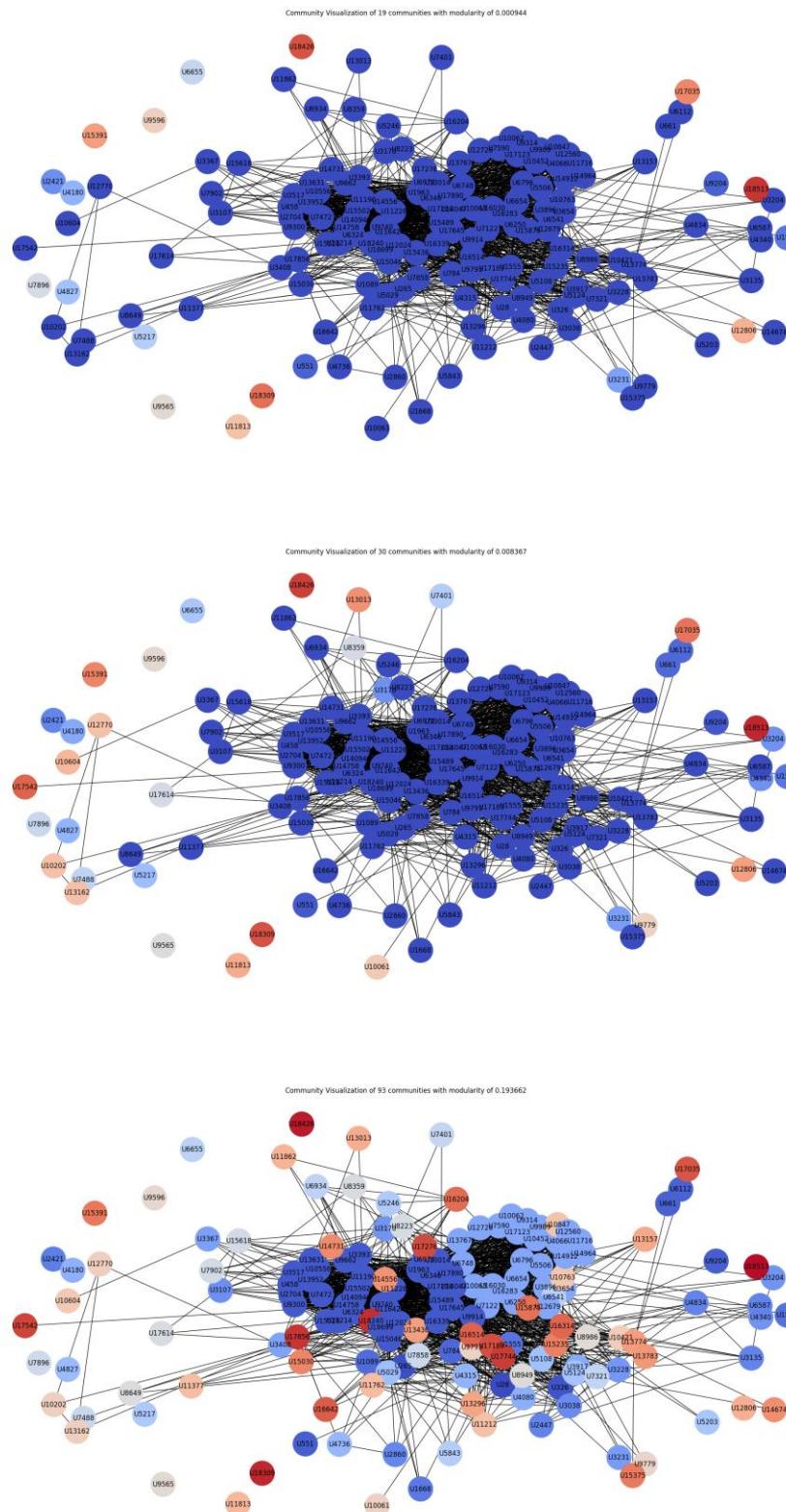
```

1 def create_community_node_colors(graph, communities):
2     num_communities = len(communities)
3     cmap = plt.colormaps['coolwarm']
4     node_colors = []
5     for node in graph:
6         for i, community in enumerate(communities):
7             if node in community:
8                 node_colors.append(cmap(i / num_communities))
9                 break
10    return node_colors
11
12 # Function to plot graph with node colors based on communities
13 def visualize_communities(graph, communities, subplot_index):
14     node_colors = create_community_node_colors(graph, communities)
15     modularity = round(nx.community.modularity(graph, communities), 6)
16     title = f"Community Visualization of {len(communities)} communities with modularity of {modularity}"
17     pos = nx.spring_layout(graph, k=0.7, iterations=50, seed=2)
18     plt.subplot(3, 1, subplot_index)
19     plt.title(title)
20     nx.draw(
21         graph, pos=pos, node_size=2000, node_color=node_colors,
22         with_labels=True, font_size=12, font_color="black",
23     )
24 fig, ax = plt.subplots(3, figsize=(30, 50))
25
26 # k = 1
27 visualize_communities(G, communities[0], 1)
28 # k = 12
29 visualize_communities(G, communities[11], 2)
30 # k = 22 (best)
31 visualize_communities(G, communities[best_k-1], 3)

```

Hình 4.4: Đoạn mã vẽ biểu đồ trực quan kết quả phân chia cộng đồng

⇒ Kết quả:



Hình 4.5: Kết quả trực quan phân chia cộng đồng theo từng giai đoạn k chính

- Xây dựng gợi ý phim cho từng user trong các cộng đồng được phân chia theo thuật toán

```

● ● ●

1 from collections import Counter
2
3 partition = {node: idx for idx, community in enumerate(communities[best_k-1]) for node in community}
4
5 user_movie_history = {}
6 for user, movie in edges:
7     if user not in user_movie_history:
8         user_movie_history[user] = []
9     user_movie_history[user].append(movie)
10
11 # Tạo danh sách phim phổ biến trong từng cộng đồng
12 community_recommendations = {}
13 for community_id in set(partition.values()):
14     # Lấy danh sách các thành viên trong cộng đồng
15     community_members = [user for user, com_id in partition.items() if com_id == community_id]
16
17     # Đếm các bộ phim đã xem trong cộng đồng
18     movie_counter = Counter()
19     for user in community_members:
20         movies = user_movie_history.get(user, [])
21         movie_counter.update(movies)
22
23     # Chọn top 5 phim phổ biến trong cộng đồng
24     popular_movies = [movie for movie, _ in movie_counter.most_common(10)]
25     community_recommendations[community_id] = popular_movies
26
27 # Tạo gợi ý cho từng người dùng
28 user_recommendations = {}
29 for user, community_id in partition.items():
30     # Lấy phim phổ biến trong cộng đồng mà người dùng chưa xem
31     movies_watched = set(user_movie_history.get(user, []))
32     recommendations = [movie for movie in community_recommendations[community_id] if movie not in movies_watched]
33     user_recommendations[user] = recommendations
34
35 #####
36 user = 'U28'
37 print(f"Recommendation for user {user} based on community:")
38 print(user_recommendations[user])

```

Hình 4.6: Đoạn mã gợi ý phim cho người dùng từ kết quả khám phá cộng đồng

⇒ Kết quả:

Recommendation for user U14094 based on community:

[614930, 747188, 335977, 814776, 603692, 493529, 447365, 575264,
 532408, 565770, 385687, 957314, 941019, 882569, 649609, 569094,
 884605, 986054, 1142216]

4.3. Louvain

4.3.1. Lý thuyết

- Thuật toán Louvain là một thuật toán đồ thị được sử dụng để phân cụm đồ thị, phân chia các đỉnh của đồ thị thành các cụm (clusters) dựa trên các liên kết giữa chúng.

$$Q = \frac{1}{2m} \sum_{i=1}^N \sum_{j=1}^N \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

Trong đó:

A_{ij}: Là trọng số của liên kết giữa đỉnh i và đỉnh j
 m: Tổng trọng số của tất cả các liên kết trong mạng
 k_i: Tổng trọng số của các liên kết ra khỏi đỉnh i
 c_i: Cộng đồng mà đỉnh i thuộc về.
 δ(c_i, c_j): Hàm delta Kronecker

4.3.2. Thực hiện trên dữ liệu

```

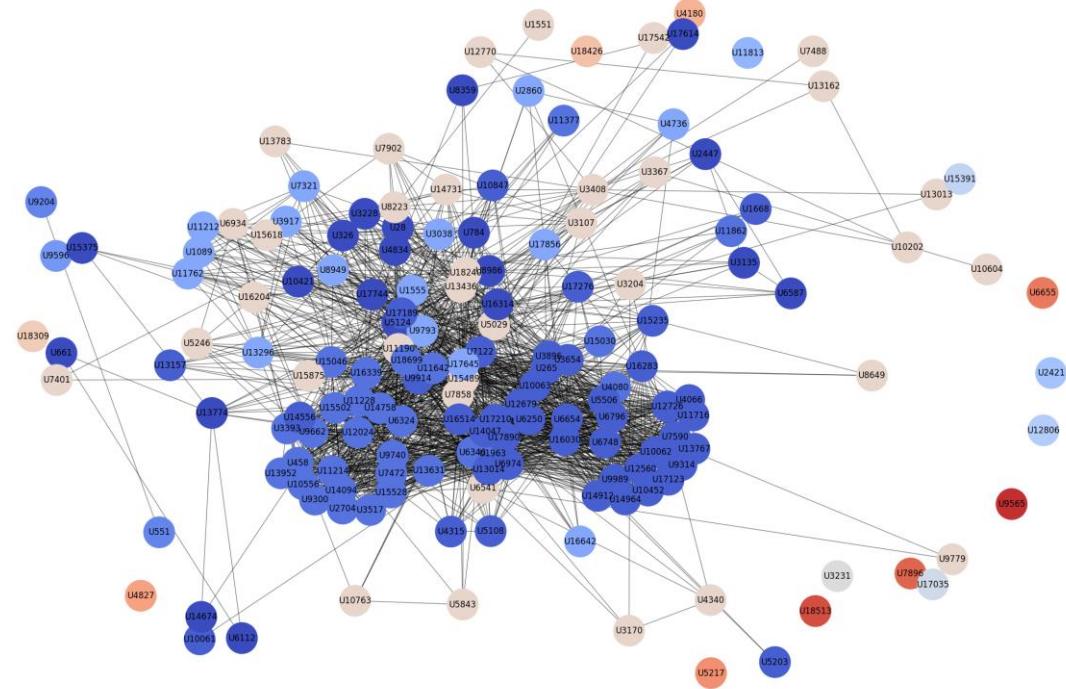
● ● ●
1 G = user_movie.copy()
2
3 # Sử dụng thuật toán Louvain để phát hiện cộng đồng
4 partition = community_louvain.best_partition(G)
5
6 # Chuyển đổi partition thành danh sách cộng đồng
7 louvain_communities = []
8 for community_id in set(partition.values()):
9   community_nodes = [node for node, comm_id in partition.items() if comm_id == community_id]
10  louvain_communities.append(community_nodes)
11
12 # Tính toán Modularity cho phân chia cộng đồng Louvain
13 modularity_value = nx.community.modularity(G, louvain_communities)
14
15 # Hiển thị cộng đồng trên biểu đồ
16 value = list(partition.values())
17
18 plt.figure(figsize=(30, 20))
19 plt.axis('off')
20 pos = nx.spring_layout(G, k=0.7)
21 cmap = plt.colormaps['coolwarm'] # Lấy bảng màu 'coolwarm'
22
23 # Tạo màu sắc cho từng nút
24 num_colors = max(value) + 1
25 colors = [cmap(i / num_colors) for i in range(num_colors)]
26
27 # Vẽ các nút và các cạnh
28 nx.draw_networkx_nodes(G, pos, node_size=2000,
29   node_color=[colors[partition[node]] for node in G.nodes()])
30 nx.draw_networkx_edges(G, pos, alpha=0.5)
31 nx.draw_networkx_labels(G, pos)
32
33 # Thêm giá trị Modularity vào tiêu đề
34 plt.title(f"Community Detection using Louvain Method with {num_colors} communities and Modularity = {modularity_value:.4f}", fontsize=20)
35
36 plt.show()

```

Hình 4.7: Đoạn mã khám phá cộng đồng bằng thuật toán Louvain

⇒ Kết quả:

Community Detection using Louvain Method with 22 communities and Modularity = 0.2938



Hình 4.8: Trực quan kết quả phân chia cộng đồng

```

● ● ●

1 user_movie_history = {}
2 for user, movie in edges:
3     if user not in user_movie_history:
4         user_movie_history[user] = []
5     user_movie_history[user].append(movie)
6
7 # Tạo danh sách phim phổ biến trong từng cộng đồng
8 community_recommendations = {}
9 for community_id in set(partition.values()):
10    # Lấy danh sách các thành viên trong cộng đồng
11    community_members = [user for user, com_id in partition.items() if com_id == community_id]
12
13    # Đếm các bộ phim đã xem trong cộng đồng
14    movie_counter = Counter()
15    for user in community_members:
16        movies = user_movie_history.get(user, [])
17        movie_counter.update(movies)
18
19    # Chọn top 5 phim phổ biến trong cộng đồng
20    popular_movies = [movie for movie, _ in movie_counter.most_common(20)]
21    community_recommendations[community_id] = popular_movies
22
23 # Tạo gợi ý cho từng người dùng
24 user_recommendations = {}
25 for user, community_id in partition.items():
26    # Lấy phim phổ biến trong cộng đồng mà người dùng chưa xem
27    movies_watched = set(user_movie_history.get(user, []))
28    recommendations = [movie for movie in community_recommendations[community_id] if movie not in movies_watched]
29    user_recommendations[user] = recommendations
30
31 #####
32 user = 'U28'
33 print(f'Recommendation for user {user} based on community:')
34 print(user_recommendations[user])

```

Hình 4.9: Đoạn mã gợi ý phim cho người dùng từ kết quả cộng đồng

⇒ Kết quả:

Recommendation for user U28 based on community:

[536554, 934433, 298618, 631842, 758323, 502356, 640146, 594767, 697843, 722149, 713704, 532408, 603692, 667538, 385687, 768362, 881164, 999644]

Phần 5. DỰ ĐOÁN LIÊN KẾT

5.1. Common Neighbors (CN)

- Chỉ số "Common Neighbors" (CN) đo lường số lượng đỉnh (node) mà hai đỉnh A và B có chung liên kết trong một đồ thị.

$$f_{CN}(A, B) = |N(A) \cap N(B)|$$

Trong đó:

$N(A)$ là tập hợp các đỉnh hàng xóm của A

$N(B)$ là tập hợp các đỉnh hàng xóm của B

$|N(A) \cap N(B)|$ là số đỉnh chung của cả hai tập hợp

- Nhận xét:

- Đơn giản và dễ hiểu, đo lường số lượng người hàng xóm chung
- Tuy nhiên, không tính đến số lượng kết nối của người hàng xóm, có thể dẫn đến sự thiên lệch khi so sánh các đỉnh với số lượng người hàng xóm khác nhau

5.2. Jaccard Coefficient (JC)

- Jaccard Coefficient: đo tỉ lệ giữa số node chung của 2 node và tổng số node mà 2 node có liên kết:

$$f_{Jaccard}(A, B) = \frac{|N(A) \cap N(B)|}{|N(A) \cup N(B)|}$$

Trong đó:

$N(A)$ và $N(B)$: Là tập hợp các đỉnh hàng xóm của đỉnh A và B

$|N(A) \cap N(B)|$: Là số lượng đỉnh chung mà cả hai đỉnh A và B có.

$|N(A) \cup N(B)|$: Là số lượng tất cả các đỉnh có trong ít nhất một trong hai tập hợp $N(A)$ hoặc $N(B)$.

- Nhận xét:
 - Cải tiến CN bằng cách sử dụng tỷ lệ giữa số người hàng xóm chung và tổng số người hàng xóm của cả hai đỉnh
 - Phương pháp này làm giảm ảnh hưởng của các đỉnh có số lượng người hàng xóm quá lớn, giúp tính toán chính xác hơn

5.3. Adamic-Adar Index (AA)

- Chỉ số Adamic-Adar (AA) là một phương pháp đo lường sự tương đồng giữa hai đỉnh trong một đồ thị dựa trên số lượng hàng xóm chung của chúng, nhưng có điều chỉnh để ưu tiên hơn cho các đỉnh có độ kết nối thấp hơn.

$$f_{AA} = \sum_{u \in N(A) \cap N(B)} \frac{1}{\log(|k_u|)}$$

Trong đó:

$N(A)$ và $N(B)$: Là tập hợp các đỉnh hàng xóm của A và B .

$N(A) \cap N(B)$: Là tập hợp đỉnh mà cả A và B đều có hàng xóm chung.

k_u : Là số lượng hàng xóm của đỉnh u

$|k_u|$: Là độ kết nối của đỉnh u , tức là số lượng đỉnh mà u có liên kết trực tiếp.

- Nhận xét:
 - Thêm trọng số cho các người hàng xóm, giảm tầm quan trọng của những người có kết nối nhiều
 - Phương pháp này hữu ích trong việc phát hiện các mối quan hệ hiếm gặp giữa các đỉnh

5.4. Preferential Attachment (PA)

- Chỉ số Preferential Attachment (PA) là một mô hình lý thuyết trong lý thuyết đồ thị, mô tả cách mà các đỉnh mới có xu hướng kết nối với các đỉnh đã có độ kết nối cao.

$$f_{PA} = |N(A)| * |N(B)|$$

Trong đó:

$N(A)$: Tập hợp các đỉnh hàng xóm của đỉnh A

$|N(A)|$: Số lượng đỉnh mà A kết nối

$N(B)$: Tập hợp các đỉnh hàng xóm của đỉnh B

$|N(B)|$: Số lượng đỉnh mà B kết nối

- Nhận xét:

- Dựa trên nguyên lý rằng các đỉnh có nhiều liên kết hơn sẽ thu hút thêm nhiều liên kết mới
- PA không trực tiếp đo lường sự tương đồng, mà mô phỏng sự phát triển tự nhiên của mạng trong các hệ thống động, ví dụ như mạng xã hội hoặc mạng Internet

5.5. Nhận xét

- Các số đo từ các thuật toán CN, JS, AA và PA càng cao thì mức độ tương đồng giữa các đỉnh càng lớn, cho thấy khả năng chúng có thể liên kết với nhau cao
 - CN: Tốt khi có nhiều hàng xóm chung
 - JC: Tốt khi cân bằng giữa hàng xóm chung và tổng số hàng xóm
 - AA: Tốt khi người hàng xóm chung có ít kết nối, làm nổi bật sự khác biệt
 - PA: Tốt khi các đỉnh có độ kết nối cao, dễ thu hút liên kết

5.6. Thực hiện trên dữ liệu



```
1 # === 2. Dự đoán liên kết bằng Heuristics ===
2 predicted_links = []
3
4 # Ngưỡng liên kết tốt
5 threshold_CN = 1
6 threshold_JC = 0.1
7 threshold_AA = 0.5
8 threshold_PA = 1.0
9
10 # Common Neighbors
11 cn_scores = []
12 for u, v in combinations(user_movie.nodes(), 2): # Tất cả các cặp user
13     if not user_movie.has_edge(u, v): # Chỉ xét cặp user chưa liên kết
14         common_neighbors = len(list(nx.common_neighbors(user_movie, u, v)))
15         if common_neighbors >= threshold_CN:
16             cn_scores.append((u, v, common_neighbors))
17
18 # Jaccard Coefficient
19 jaccard_scores = list(nx.jaccard_coefficient(user_movie))
20 for u, v, score in jaccard_scores:
21     if score >= threshold_JC:
22         predicted_links.append((u, v, "Jaccard Coefficient", score))
23
24 # Adamic-Adar Index
25 adamic_adar_scores = list(nx.adamic_adar_index(user_movie))
26 for u, v, score in adamic_adar_scores:
27     if score >= threshold_AA:
28         predicted_links.append((u, v, "Adamic-Adar Index", score))
29
30 # Preferential Attachment
31 pa_scores = list(nx.preferential_attachment(user_movie))
32 for u, v, score in pa_scores:
33     if score >= threshold_PA:
34         predicted_links.append((u, v, "Preferential Attachment", score))
```



```
1 # === 3. Gợi ý phim cho user dựa trên liên kết mạnh ===
2 recommendations = {}
3
4 # Tạo dictionary lưu phim mà mỗi user đã xem
5 user_to_movies = ratings.groupby('userId')['tmdbId'].apply(set).to_dict()
6
7 for u, v, method, score in predicted_links:
8     # Khởi tạo danh sách gợi ý nếu chưa có
9     if u not in recommendations:
10         recommendations[u] = set()
11     if v not in recommendations:
12         recommendations[v] = set()
13
14     # Gợi ý phim từ user này cho user kia
15     movies_u = user_to_movies.get(u, set())
16     movies_v = user_to_movies.get(v, set())
17
18     recommendations[u].update(movies_v - movies_u) # Gợi ý phim của v cho u
19     recommendations[v].update(movies_u - movies_v) # Gợi ý phim của u cho v
20
21 # === 4. Lọc top n bộ phim được gợi ý nhiều nhất cho mỗi user ===
22 top_n = 20 # Số lượng phim gợi ý tối đa cho mỗi user
23
24 # Lưu trữ kết quả gợi ý top n phim cho mỗi user
25 top_recommendations = {}
26
27 # Đếm số lần mỗi bộ phim được gợi ý cho mỗi user
28 for user, movies in recommendations.items():
29     movie_counts = Counter(movies) # Đếm số lần mỗi phim xuất hiện
30     top_movies = movie_counts.most_common(top_n) # Lấy top n phim
31     top_recommendations[user] = [movie for movie, _ in top_movies]
32
33 # === 5. Kết quả ===
34 print("\nGợi ý phim cho từng user (Top n gợi ý):")
35 for user, top_movies in top_recommendations.items():
36     print(f"{user}: {top_movies}")
37
```

Hình 5.1: Đoạn mã thực hiện dự đoán liên kết và gợi ý phim từ kết quả

⇒ Kết quả:

Gọi ý phim cho từng user (Top n gợi ý):

```

U340: [736769, 635910, 1015303, 565770, 881164, 912908, 915980, 614930, 800787, 880150, 796185, 1000475, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323, 920125]
U13157: [736769, 635910, 1015303, 980489, 565770, 881164, 915980, 614930, 800787, 880150, 796185, 1000475, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323, 920125]
U9779: [736769, 635910, 1015303, 565770, 881164, 912908, 915980, 614930, 800787, 880150, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 920125, 737853, 945729]
U14758: [736769, 635910, 1015303, 980489, 565770, 912908, 881164, 614930, 800787, 796185, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323, 920125, 762430, 737853]
U5006: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 737853, 762430, 945729, 915523, 346698]
U12679: [736769, 635910, 1015303, 565770, 881164, 636173, 1134865, 614930, 1016084, 555285, 796185, 1000475, 985883, 937249, 631842, 934433, 960292, 1121575, 1148969]
U13436: [569094, 635910, 1015303, 980489, 913673, 565770, 636173, 614930, 800787, 1016084, 555285, 878361, 796185, 575264, 937249, 631842, 934433, 960292, 1148969, 353577]
U16030: [736769, 635910, 569094, 980489, 565770, 913673, 881164, 636173, 1134865, 614930, 800787, 1016084, 555285, 796185, 985883, 1000475, 937249, 934433, 631842, 960292]
U28: [736769, 635910, 1015303, 980489, 565770, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323, 920125, 737853, 762430, 945729]
U11716: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 920125, 737853, 762430, 945729, 915523, 346698]
U18240: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 555285, 878361, 1000475, 575264, 937249, 631842, 934433, 960292, 1121575, 353577, 961323, 603692]
U9314: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 737853, 762430, 945729, 915523, 346698]
U4080: [736769, 635910, 1015303, 565770, 881164, 912908, 614930, 800787, 796185, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323, 737853, 920125, 945729, 915523]
U784: [736769, 569094, 635910, 1015303, 980489, 565770, 636173, 1134865, 614930, 581395, 1016084, 878361, 796185, 985883, 575264, 937249, 934433, 631842, 960292, 1148969]
U13767: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 737853, 762430, 945729, 915523, 346698]
U4066: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 737853, 762430, 945729, 915523, 346698]
U12726: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 737853, 762430, 945729, 915523, 346698]
U13296: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323, 737853, 920125, 762430, 945729]
U16314: [736769, 635910, 1015303, 980489, 565770, 1134865, 614930, 1016084, 555285, 878361, 1000475, 575264, 937249, 934433, 631842, 960292, 1121575, 353577, 961323, 616747]
U17189: [736769, 569094, 635910, 1015303, 980489, 565770, 1134865, 614930, 1016084, 555285, 878361, 1000475, 985883, 937249, 631842, 934433, 960292, 1121575, 353577, 961323]
U9793: [736769, 635910, 1015303, 980489, 565770, 881164, 1134865, 614930, 1016084, 555285, 878361, 1000475, 575264, 937249, 934433, 631842, 960292, 1121575, 353577, 616747]
U16514: [736769, 635910, 1015303, 980489, 565770, 881164, 614930, 800787, 796185, 1000475, 934433, 631842, 1148969, 603692, 758323, 920125, 737853, 762430, 945729, 915523]
U5029: [736769, 635910, 569094, 1015303, 980489, 565770, 912908, 636173, 881164, 1134865, 614930, 581395, 1016084, 555285, 796185, 878361, 985883, 575264, 937249, 631842]
...
U661: [736769, 635910, 1015303, 980489, 565770, 881164, 912908, 915980, 614930, 800787, 880150, 796185, 1000475, 631842, 1148969, 603692, 1151534, 1158706, 758323, 920125]
U10604: [736769, 635910, 1015303, 980489, 565770, 881164, 912908, 915980, 614930, 800787, 880150, 796185, 1000475, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323]
U9204: [736769, 635910, 1015303, 980489, 565770, 881164, 912908, 915980, 614930, 800787, 880150, 796185, 1000475, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323]
U551: [736769, 635910, 1015303, 980489, 565770, 881164, 912908, 915980, 614930, 800787, 880150, 796185, 1000475, 934433, 631842, 1148969, 603692, 1151534, 1158706, 758323]
```

Hình 5.2: Kết quả ứng dụng dự đoán liên kết vào gợi ý phim cho người dùng

Phần 6. MÔ HÌNH LAN TRUYỀN THÔNG TIN

6.1. Mô hình IC

- Mô hình Independent Cascade (IC) là một mô hình lý thuyết được sử dụng trong nghiên cứu lan truyền thông tin hoặc ảnh hưởng trong mạng xã hội. Mô hình này giúp mô phỏng cách mà một thông điệp, ý tưởng, hoặc một hiện tượng có thể lan truyền từ một nhóm người (đỉnh) sang nhóm khác trong một mạng.
- Trong mô hình Independent Cascade, mỗi đỉnh trong mạng có thể "lây lan" trạng thái (như trạng thái bị nhiễm hoặc không nhiễm) cho các đỉnh hàng xóm của nó với một xác suất nhất định. Quá trình này diễn ra theo các bước sau:
 - Khởi đầu: Một tập hợp các đỉnh được chọn để bắt đầu lan truyền (thường được gọi là các "đỉnh khởi đầu").
 - Lan truyền: Mỗi đỉnh đã bị nhiễm sẽ cố gắng lây lan trạng thái của mình cho các đỉnh hàng xóm chưa bị nhiễm.
 - Xác suất lây lan: Với mỗi đỉnh hàng xóm, xác suất để nó bị nhiễm từ đỉnh lây lan được xác định trước và là độc lập với các đỉnh khác.

6.2. Thực hiện trên dữ liệu

- Chọn ngẫu nhiên 5 user để lan truyền thông tin theo mô hình IC, kiểm tra số nút bị ảnh hưởng khi lan truyền.

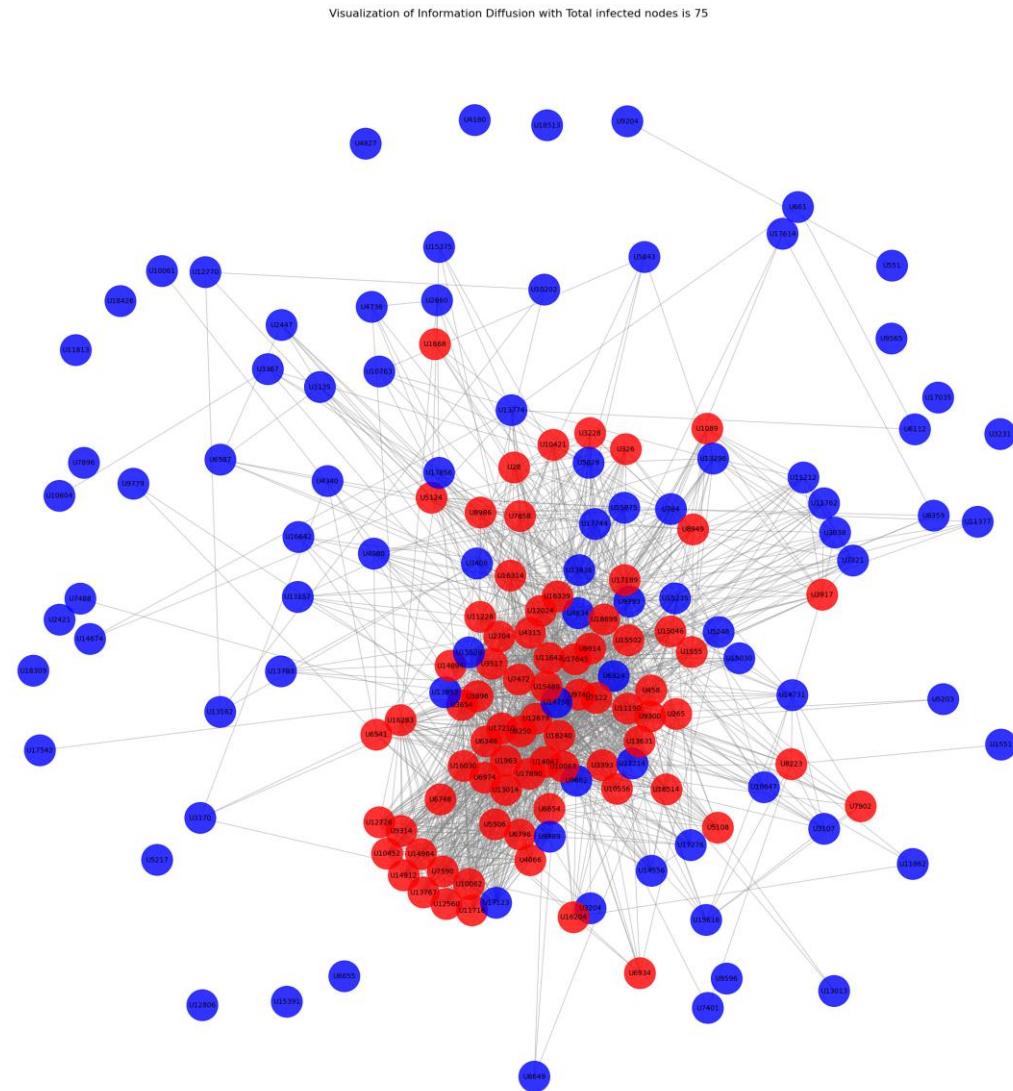
```

● ● ●

1 # Khởi tạo nút nguồn lan truyền
2 initial_nodes = random.sample(list(user_movie.nodes()), k=5)
3
4 # Mô phỏng lan truyền thông tin
5 infected_nodes = independent_cascade(user_movie, initial_nodes, p=0.05)
6
7 # -----
8 # VIZUALIZATION
9 #
10 # Xác định vị trí của các nút bằng cách sử dụng layout (spring layout được chọn để dễ nhìn)
11 pos = nx.spring_layout(user_movie, k=0.7) # Cố định layout để đồ thị không thay đổi mỗi lần vẽ
12
13 # Vẽ đồ thị
14 plt.figure(figsize=(30, 30)) # Kích thước đồ thị
15
16 # Tô màu các nút: nút bị lây nhiễm là đỏ, nút khác là xanh
17 node_colors = ['red' if node in infected_nodes else 'blue' for node in user_movie.nodes()] # Sử dụng infected_nodes
18
19 # Vẽ các nút
20 nx.draw_networkx_nodes(
21     user_movie,
22     pos,
23     node_color=node_colors,
24     node_size=2000, # Tăng kích thước nút
25     alpha=0.8       # Làm nút trong suốt một chút để dễ quan sát
26 )
27
28 # Vẽ các cạnh
29 nx.draw_networkx_edges(
30     user_movie,
31     pos,
32     edge_color="gray",
33     alpha=0.5, # Độ trong suốt của cạnh
34     width=1      # Độ dày cạnh
35 )
36
37 # Thêm nhãn vào các nút
38 nx.draw_networkx_labels(
39     user_movie,
40     pos,
41     font_size=10, # Kích thước chữ
42     font_color="black"
43 )
44
45 # Tiêu đề và hiển thị đồ thị
46 plt.title(f"Visualization of Information Diffusion with Total infected nodes is {len(infected_nodes)} ", fontsize=16)
47 plt.axis("off") # Tắt trục
48 plt.show()
49

```

Hình 6.1: Đoạn mã hiện thực mô hình lan truyền thông tin IC



Hình 6.2: Trực quan kết quả các đỉnh bị ảnh hưởng bởi lan truyền

- Từ mô hình lan truyền, thực hiện tạo danh sách đề xuất phim cho người bị ảnh hưởng



```

1  def information_diffusion_recommendations(user_user, edges, top_n=20):
2      # Tạo một dictionary để lưu trữ kết quả gợi ý phim cho mỗi người dùng
3      top_recommendations = {}
4
5      # Chọn ngẫu nhiên một số người dùng làm nguồn lan truyền
6      initial_nodes = random.sample(list(user_user.nodes()), k=10)
7
8      if isinstance(edges, list):
9          edges = pd.DataFrame(edges, columns=["userId", "tmdbId"])
10
11     # Mô phỏng lan truyền thông tin
12     infected_nodes = independent_cascade(user_user, initial_nodes, p=0.05)
13
14     edges['userId'] = edges['userId'].astype(str)
15     # Duyệt qua các nút bị lây nhiễm và gợi ý phim
16     for node in infected_nodes:
17         # Lấy các phim mà người dùng bị lây nhiễm đã xem
18         user_rated_movies = edges[edges['userId'] == node]['tmdbId']
19
20         # Tạo danh sách phim được đề xuất cho người dùng này
21         recommended_movies = []
22
23         # Duyệt qua các người dùng bị lây nhiễm khác để gợi ý phim
24         for infected_user in infected_nodes:
25             if infected_user != node: # Không gợi ý phim đã xem của chính người dùng
26                 user_rated_movies = edges[edges['userId'] == infected_user]['tmdbId']
27                 recommended_movies.extend(user_rated_movies)
28
29             # Loại bỏ các phim mà người dùng đã xem
30             recommended_movies = list(set(recommended_movies) - set(user_rated_movies)) # Loại bỏ trùng
31             recommended_movies = recommended_movies[:top_n] # Lấy top N phim
32
33             # Lưu kết quả gợi ý cho người dùng vào dictionary
34             top_recommendations[node] = recommended_movies
35
36     return top_recommendations
37
38 # Gọi hàm để lấy gợi ý phim
39 top_recommendations = information_diffusion_recommendations(user_movie, ratings)
40
41 # In ra gợi ý phim cho từng người dùng dựa trên lan truyền thông tin
42 print("\nGợi ý phim cho từng user (Top n gợi ý):")
43 for user, top_movies in top_recommendations.items():
44     print(f"{user}: {top_movies}")
45

```

Hình 6.3: Đoạn mã gợi ý phim từ các đỉnh bị ảnh hưởng bởi lan truyền

Mạng xã hội

Hình 6.4: Kết quả gợi ý phim từ quá trình lan truyền và gợi ý

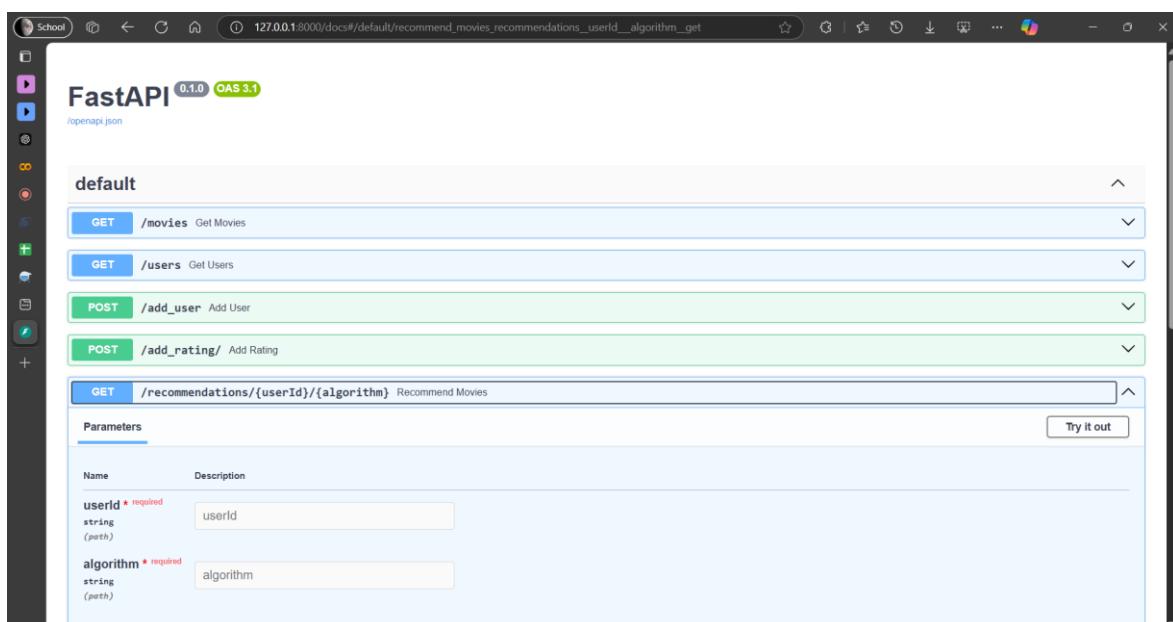
Phần 7. WEBSITE

7.1. Công nghệ sử dụng

- **API:** Thư viện FastAPI của Python
- **Giao diện Webite:** Thư viện Streamlit của Python

7.2. Cài đặt API

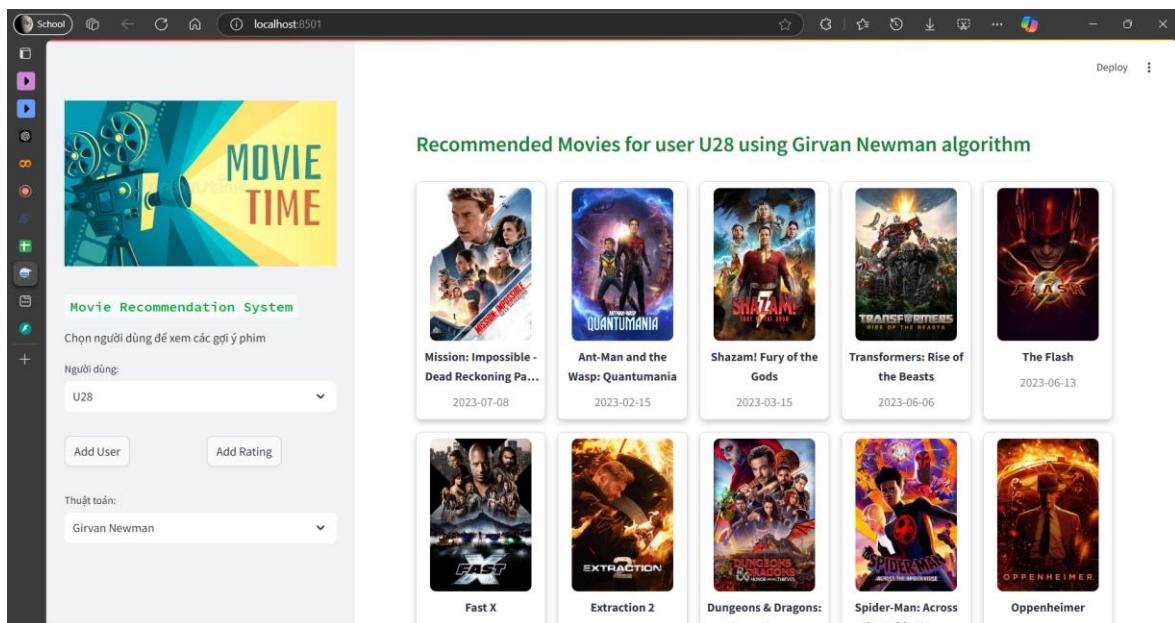
- Từ các thuật toán phát hiện cộng đồng, dự đoán liên kết và mô hình lan truyền dữ liệu đã thực nghiệm với bộ dữ liệu, nhóm sử dụng FastAPI để triển khai các mô hình thành API để có thể sử dụng.
- API sẽ đọc dữ liệu từ file dataset để sử dụng, khi được khởi động API sẽ dùng dữ liệu đó để gợi ý phim cho các user và lưu vào biến tạm để có thể trả về kết quả gợi ý phim cho từng user. Ngoài ra API còn có chức năng thêm user mới và thêm đánh giá mới. Trong trường hợp thêm đánh giá, các mô hình sẽ tự động thực hiện tại để gợi ý lại phim cho user.



Hình 7.1: Giao diện API được triển khai trên host

7.3. Giao diện Website

Giao diện Website được xây dựng bằng thư viện Streamlit cho phép người sử dụng có thể xem kết quả gợi ý phim của từng user theo từng mô hình gợi ý khác nhau, kết quả sẽ được trả về từ các truy vấn đến API đang được chạy trên một host riêng.



Hình 7.2: Giao diện tương tác trang web được triển khai trên host



Hình 7.3: Chọn thuật toán, mô hình để gợi ý

PHÂN CÔNG CÔNG VIỆC

Công việc	Lê Thị Lệ Trúc (Nhóm trưởng)	Hồ Quang Lâm (Thành viên)	Trần Thị Kim Anh (Thành viên)	Lê Minh Chánh (Thành viên)
Mô tả dữ liệu				X
Các đồ thị cơ bản		X		
Các thuật toán độ đo trung tâm	X		X	
Hàm đánh giá thuật toán				X
Girvan Newman	X			
Dự đoán liên kết		X		X
Mô hình lan truyền thông tin	X		X	
Website				X
Tổng kết	X			
Làm báo cáo	X	X	X	X
Làm powerpoint	X	X	X	X
Thuyết trình	X	X	X	X
Mức độ hoàn thành	100%	100%	100%	100%

TÀI LIỆU THAM KHẢO

- [1] *Thuật toán Louvain.* (n.d.). Studocu. Retrieved November 7, 2024, from <https://www.studocu.com/vn/document/truong-dai-hoc-kinh-te-thanh-pho-ho-chi-minh/toan-cao-cap/thuat-toan-louvain/57978003>
- [2] (N.d.). Sciencedirect.com. Retrieved November 7, 2024, from <https://www.sciencedirect.com/topics/computer-science/degree-distribution>
- [3] Học, L. V. T. S. Ī. (n.d.). *MỘT SỐ THUẬT TOÁN TÌM KIẾM CÔNG ĐỒNG MẠNG THÔNG QUA TỐI ƯU HÓA HÀM MODULARITY.* Edu.Vn. Retrieved December 18, 2024, from <https://gust.edu.vn/media/30/uftai-ve-tai-day30715.pdf>
- [4] Đinh, L. (2023, February 1). *P4. (Lại là) Machine Learning with Graphs.* Viblo. <https://viblo.asia/p/p4-lai-la-machine-learning-with-graphs-WR5JRm7n4Gv>

Link Github dự án: <https://github.com/chanhlm/SocialNetworks-IS353.P12.HTCL>: Project: Building a movie rating recommendation system for users based on social network algorithms