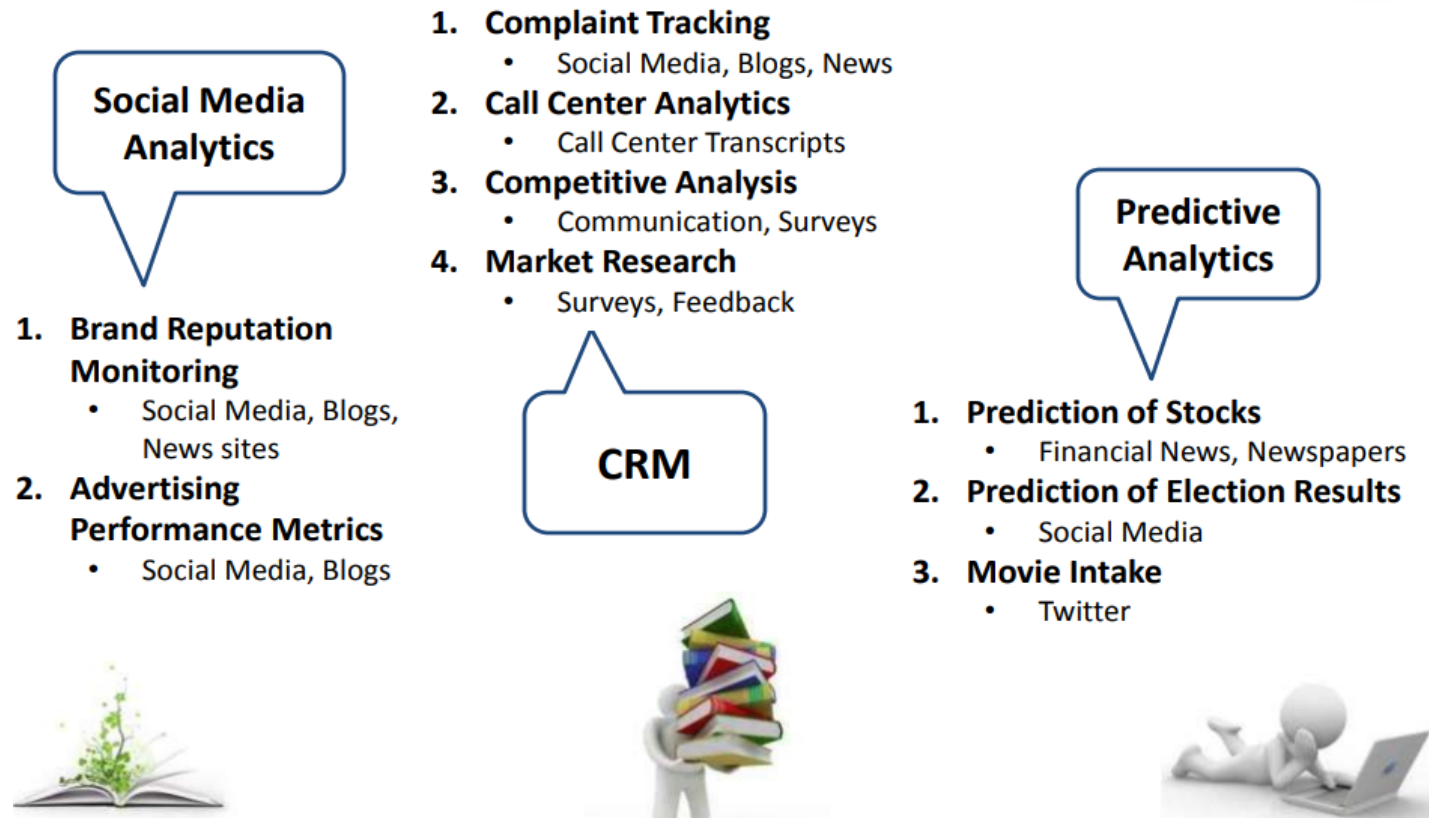


# Network Data Analytics

# Last Week Recap


## ❖ Textual Data Analytics Application



# Last Week Recap

## ❖ Characteristics of textual data:

- Unstructured
- Building blocks are words
  - Words are not independent
- Each text segment (e.g. sentence) encapsulates semantics behind



**Syntactical** Analysis: transform unstructured text to structured representation



**Semantic** Analysis

# Last Week Recap:

## Syntactical Analysis

- ❖ Transform a textual data into a multi-dimensional vector
- ❖ **Approaches:**
  - Feature Engineering: **hand-craft** the features (e.g. TF-IDF)
  - Representation learning: **auto-learn** the features (e.g. neural embedding)
- ❖ **Applications**
  - Information retrieval: search relevant documents given a query
  - Classification: categorize a text into a pre-defined label
    - e.g. spam email detection, Gmail tabbed categories
  - ...

# Last Week Recap:

## Sentiment Analysis

- ❖ Computational study of opinions, sentiments, etc., expressed in text.
  - E.g. extract from text **how people feel** about different products (Reviews, blogs, discussions, news, comments, feedback, ...)
- ❖ Techniques:
  - Classification approach
  - Lexicon approach

# 3803ICT course structure

**W1.** Introduction to Data Analytics

Data Preparation and Preprocessing

**W2.** Data Preparation and Preprocessing

Data Analysis and Interpretation

**W3.** Exploratory Data Analytics

**W4.** Statistical Data Analytics

**W5.** Predictive Data Analytics

Visualization

**W6.** Data Visualization

Analysis of special types of data

**W7.** Time Series

**W8.** Textual Data

**W9.** Graph Data

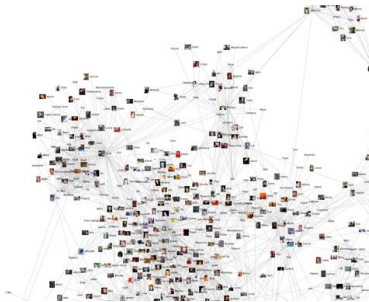
Analysis with big data infrastructure

**W10.** Distributed Data Analysis

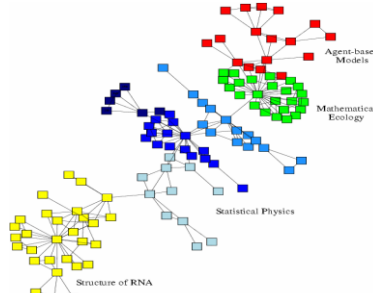
**W11.** Cloud-based Data Analysis

**W12.** Revision

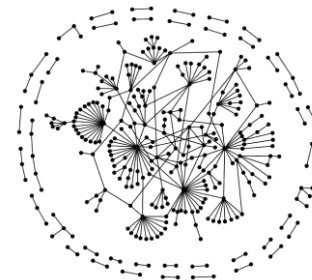
# Many Data are Networks



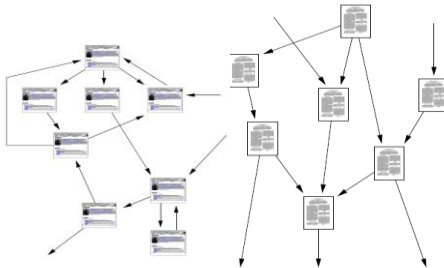
Social networks



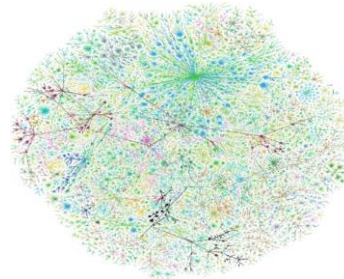
Economic networks



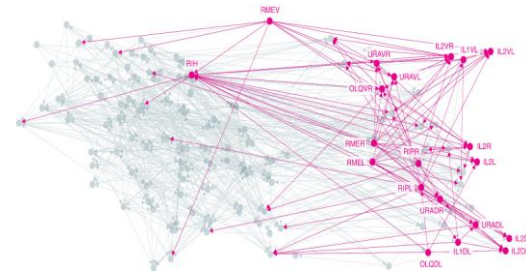
Biomedical networks



Information networks:  
Web & citations



Internet



Networks of neurons

# Network Data Analytics

- I. Network Representation
- II. Centrality Analysis
- III. Community Analysis
- IV. Information Diffusion Analysis



# I. Network Representation

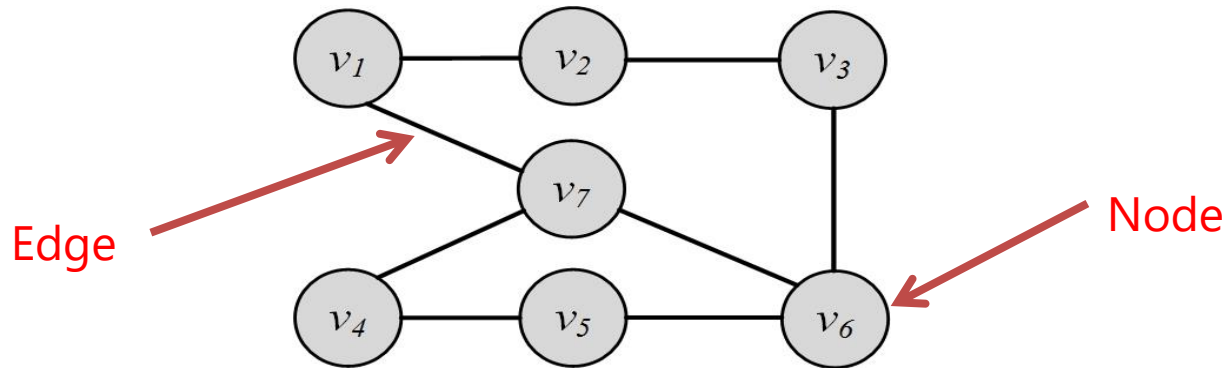
A network is a graph, or a collection of points connected by lines

❖ Points are referred to as **nodes**, **actors** or **vertices**

➤ Each node can have content (e.g. tweets in Twitter network)

❖ Connections are referred to as **edges**, **links** or **ties**

➤ Each edge can have a weight (e.g. similarity between two tweets)



$$V = \{v_1, v_2, \dots, v_n\}$$

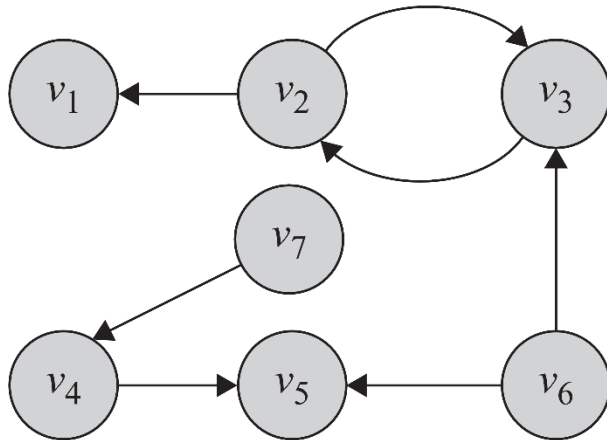
$$|V| = \mathbf{n}$$

$$E = \{e_1, e_2, \dots, e_m\}$$

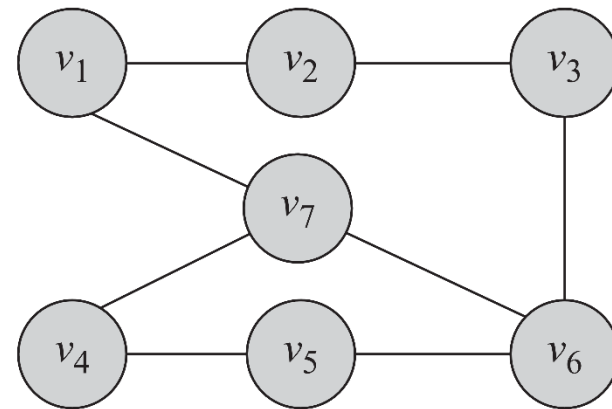
$$|E| = \mathbf{m}$$

# Directed Edges and Directed Graphs

- ❖ Edges can have directions. A directed edge is sometimes called an **arc**



(a) Directed Graph

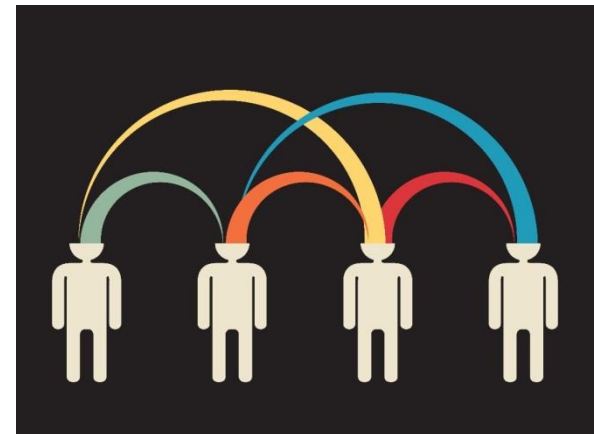
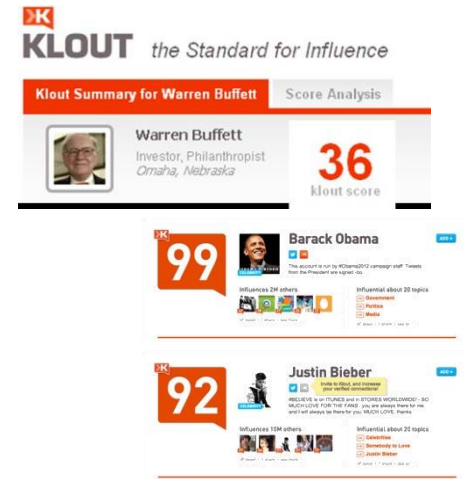


(b) Undirected Graph

- ❖ Edges are represented using their end-points  $e(v_2, v_1)$  or just  $(v_2, v_1)$
- ❖ In undirected graphs the order does not matter

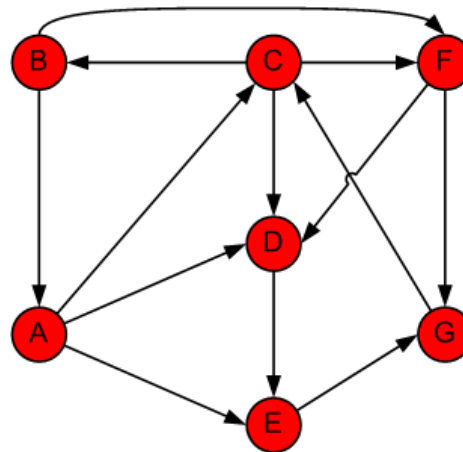
# II. Centrality

- ❖ What is centrality?
  - Centrality defines **how important** an actor is within a network
- ❖ Why centrality? a measure of **influence**
  - The act or power of producing an effect without apparent exertion of force or direct exercise of command



# Centrality: Definition

- ❖ Identify the central figures (influential individuals) in the network
  - Question: who is the most important?

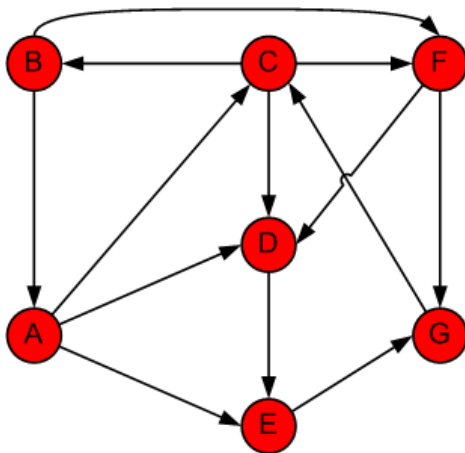


# Degree Centrality

- ❖ **Question:** who is the most important?
- ❖ **Degree centrality (DC):** ranks nodes with **more connections** higher in terms of centrality

$$C_d(v_i) = d_i$$

➤ where  $d_i$  is the number of neighbors (count both incoming and outgoing edges)



Rank

Node	DC	Rank
A	4	2
B	3	3
C	5	1
D	4	2
E	3	3
F	4	2
G	3	3

- ❖ **Shortcoming:** having more friends **does not guarantee** that someone is more important?

# Eigenvector Centrality

❖ **Principle:** Having more **important friends** provides a stronger signal

❖ **Eigenvector centrality**  $C_e(v_i)$ :

➤ We would like  $C_e(v_i)$  to be higher when important neighbors (i.e. node  $v_j$  with high  $C_e(v_j)$ ) point to  $v_i$

→ An iterative computation: stop upon convergence

$$C_e(v_i) = \frac{1}{\lambda} \sum_{j=1}^n A_{j,i} \times C_e(v_j)$$

▪ where  $\lambda$  is a normalization factor to avoid numerical overflow

❖ **Shortcoming:** In directed graphs, once a node has a high centrality, it passes all its centrality along all of its out-links.

➤ This is not desirable: not everyone known by a well-known person is well-known

➤ e.g. if you are referred by a well-known person who writes too many reference letters, then you are not too important

# PageRank Centrality

## ❖ Principle:

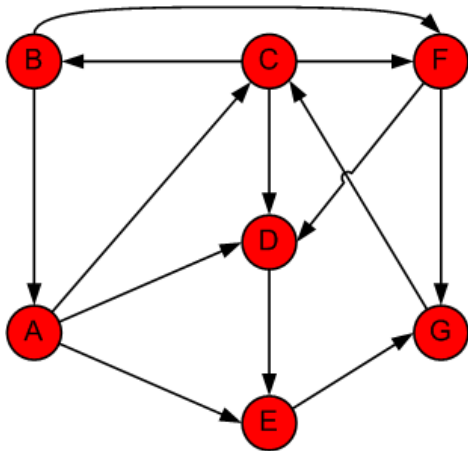
- Divide the value of passed centrality by the number of outgoing links
- Each connected neighbor only gets a fraction of the source node's centrality

$$C_p(v_i) = \frac{1}{\lambda} \sum_{j=1}^n A_{j,i} \times \frac{C_p(v_j)}{d_j^{out}}$$

- where  $\lambda$  is a normalization factor to avoid numerical overflow

# PageRank Centrality (PC)

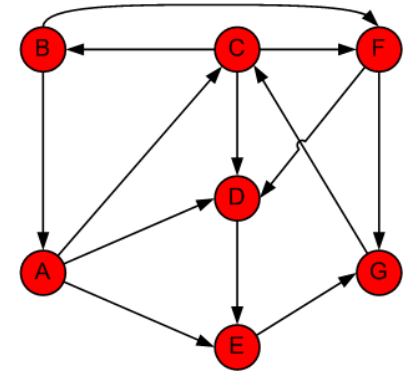
## ❖ Example



Node	PC	Rank
A	?	?
B	?	?
C	?	?
D	?	?
E	?	?
F	?	?
G	?	?

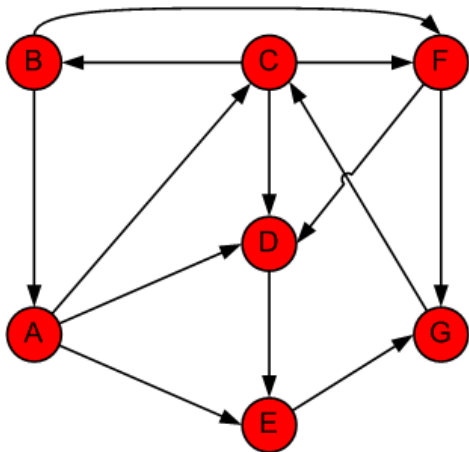


# Page Rank Centrality



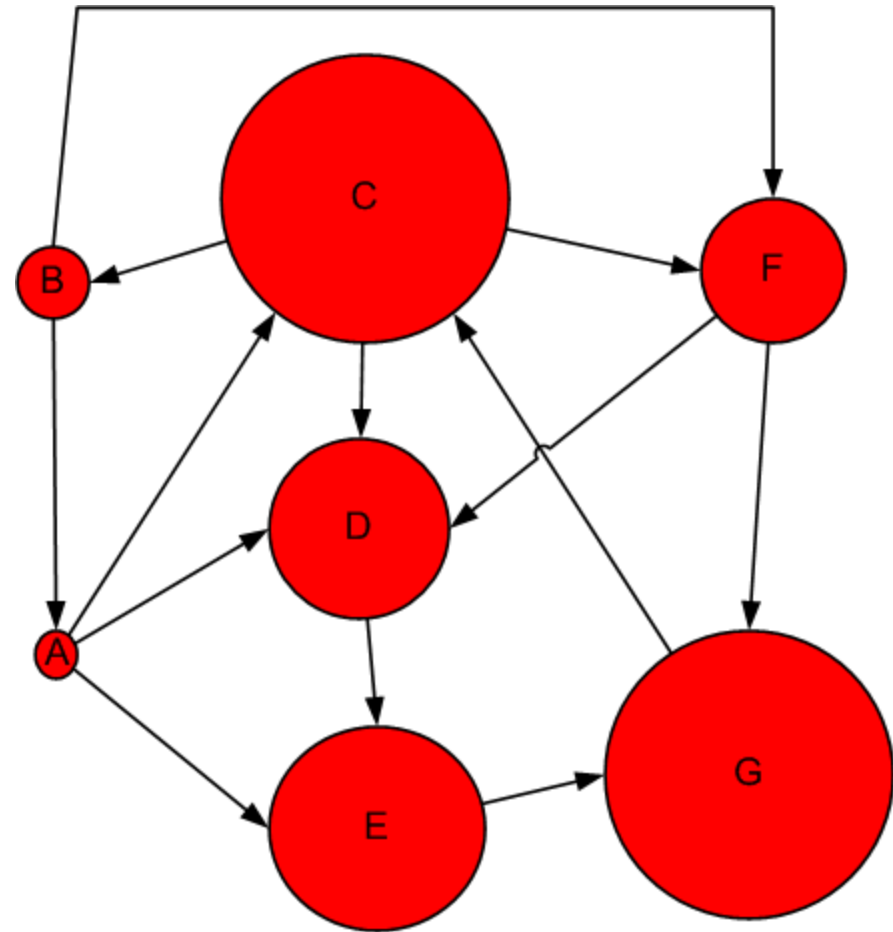
Step	A	B	C	D	E	F	G
1	0.143	0.143	0.143	0.143	0.143	0.143	0.143
2	0.071	0.048	0.190	0.167	0.190	0.119	0.214
3	0.024	0.063	0.238	0.147	0.190	0.087	0.250
4	0.032	0.079	0.258	0.131	0.155	0.111	0.234
5	0.040	0.086	0.245	0.152	0.142	0.126	0.210
6	0.043	0.082	0.224	0.158	0.165	0.125	0.204
7	0.041	0.075	0.219	0.151	0.172	0.115	0.228
8	0.037	0.073	0.241	0.144	0.165	0.110	0.230
9	0.036	0.080	0.242	0.148	0.157	0.117	0.220
10	0.040	0.081	0.232	0.151	0.160	0.121	0.215
11	0.040	0.077	0.228	0.151	0.165	0.118	0.220
12	0.039	0.076	0.234	0.148	0.165	0.115	0.223
13	0.038	0.078	0.236	0.148	0.161	0.116	0.222
14	0.039	0.079	0.235	0.149	0.161	0.118	0.219
15	0.039	0.078	0.232	0.150	0.162	0.118	0.220
<b>Rank</b>	<b>7</b>	<b>6</b>	<b>1</b>	<b>4</b>	<b>3</b>	<b>5</b>	<b>2</b>

# Effect of PageRank



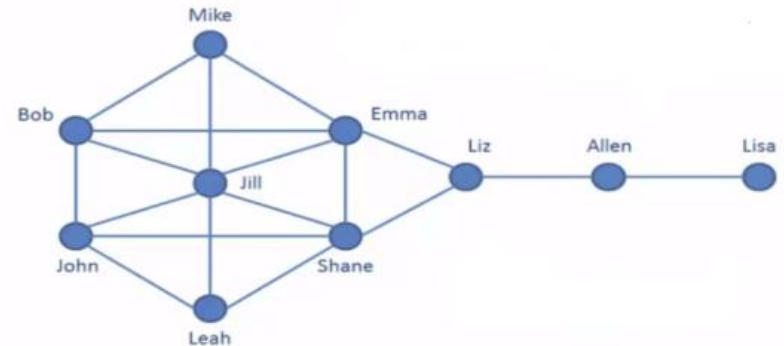
**PageRank**

Node	Rank
A	7
B	6
C	1
D	4
E	3
F	5
G	2



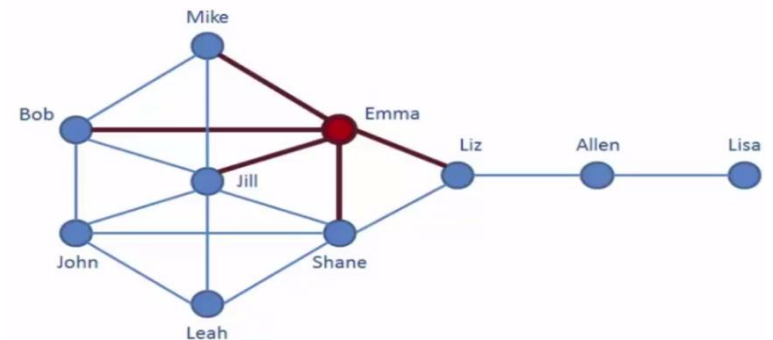
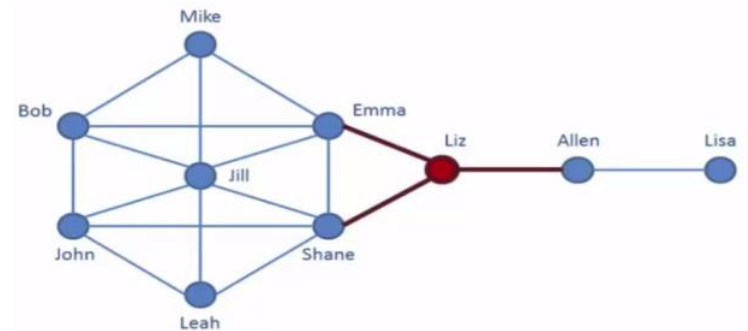
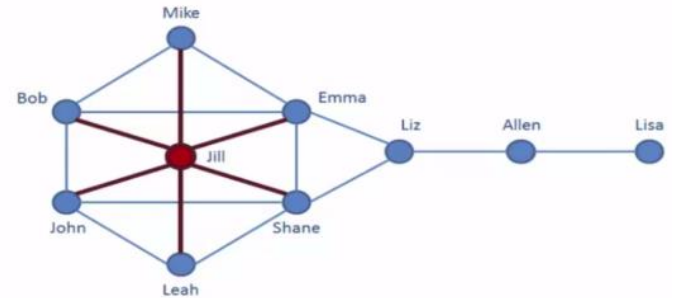
# Other types of centrality (OPTIONAL)

1. Centrality in terms of those who you are connected to
  - e.g. degree centrality, eigenvector centrality, Pagerank centrality
2. Centrality in terms of how you connect others
  - e.g. betweenness centrality
3. Centrality in terms of how fast you can reach others
  - e.g. closeness centrality



# Other types of centrality (OPTIONAL)

1. Centrality in terms of those who you are connected to
  - e.g. degree centrality, eigenvector centrality, Pagerank centrality
2. Centrality in terms of how you connect others
  - e.g. betweenness centrality
3. Centrality in terms of how fast you can reach others
  - e.g. closeness centrality



# III. Community Analysis



**[real-world] community**

A group of individuals with common *economic, social, or political* interests or characteristics, often living in *relative proximity*

# Why analyze communities?

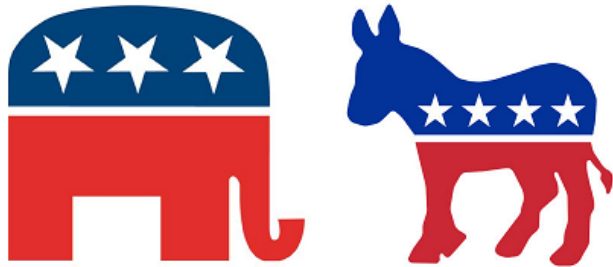
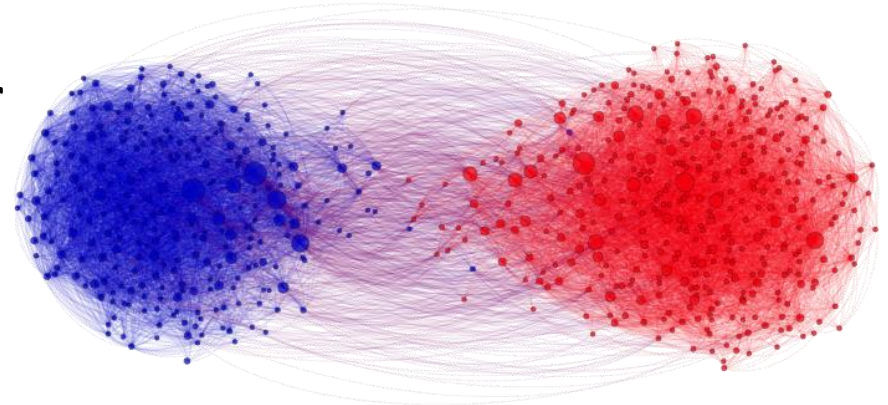


Groups provide a clear global view of user interactions

- E.g., find polarization

Analyzing communities helps better understand users

- Users form groups based on their interests



Some behaviors are only observable in a group setting and not on an individual level

- Some republican can **agree** with some democrats, but their parties can **disagree**

# Applications of Community Analysis

## ❖ Communities for Recommender Systems

### ➤ group recommendation

- Knowing individual ratings of A,B,C. What to recommend to the group?

## ❖ Communities for Digital Marketing

### ➤ group-specific marketing (e.g., selling products for housewives)

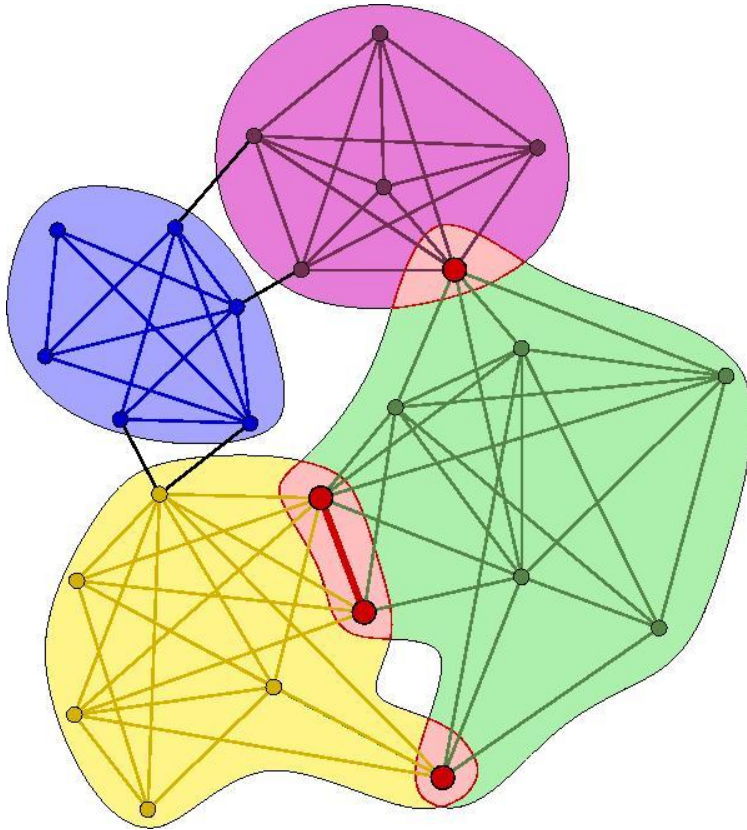
## ❖ Communities for User Behavior Analytics

### ➤ collective behavior (e.g., gossip)

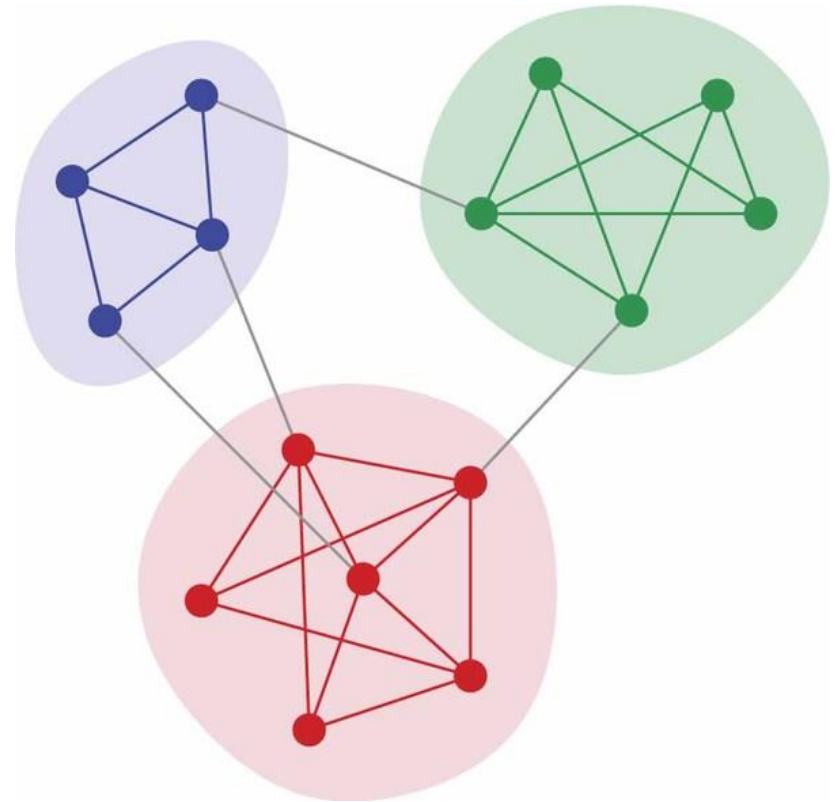




# Overlapping vs. Disjoint Communities



Overlapping Communities



Disjoint Communities



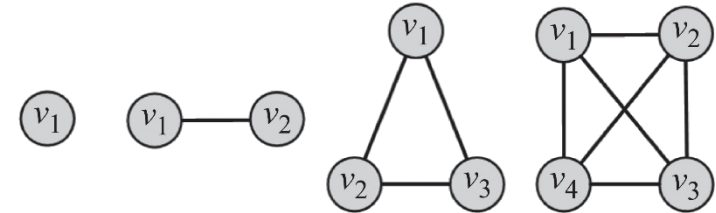
# Community analysis

- ❖ The process of finding clusters of nodes
  - With **Strong** internal connections and
  - **Weak** connections between different communities
- ❖ Ideal decomposition of a large graph:
  - Completely disjoint communities
  - There are no interactions between different communities.
- ❖ In practice,
  - find community partitions that are maximally decoupled.

# Community Detection Algorithm

**Most common subgraph searched for:**

- ❖ **Clique:** a maximum complete subgraph in which all nodes inside the subgraph are adjacent to each other



Find communities by searching for

1. **The maximum clique:** the one with the largest number of vertices, or
2. **All maximal cliques:** cliques that are not subgraphs of a larger clique; i.e., cannot be further expanded

**To overcome this, we can**

- Use Brute Force
- Relax cliques
- Use cliques as the core for larger communities

**Both problems are NP-hard**

# Clique Percolation Method (CPM)

## Clique Percolation Method (CPM)

- Uses cliques as seeds to find larger communities
- CPM finds overlapping communities

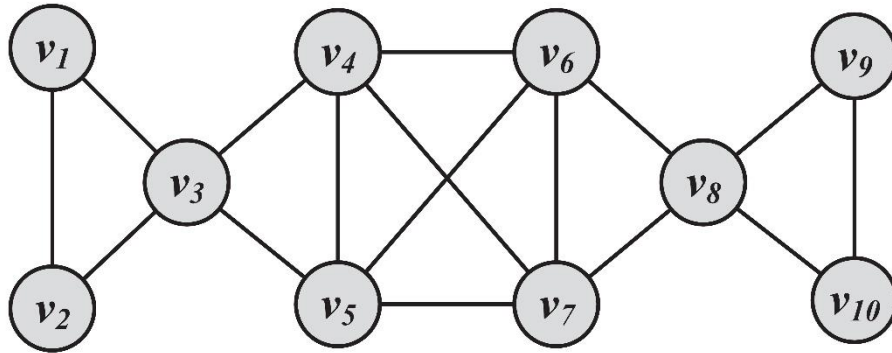
### ❖ Input

- A parameter  $k$ , and a network

### ❖ Procedure

1. Find all **cliques of size  $k$**  in the given network
2. Construct a clique graph.
  - Two cliques are **adjacent** if they **share  $k - 1$**  nodes
3. Each **connected component** in the clique graph forms a community

# Community Detection: Example



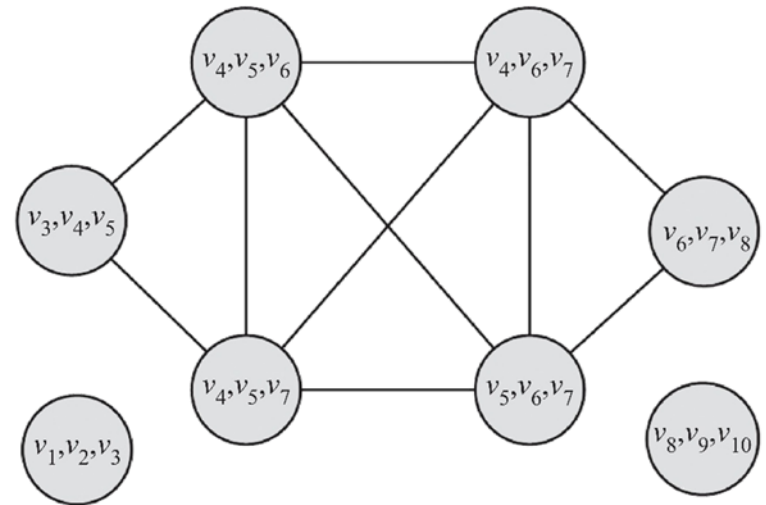
(a) Graph

## Cliques of size 3:

$\{v_1, v_2, v_3\}, \{v_3, v_4, v_5\},$   
 $\{v_4, v_5, v_6\}, \{v_4, v_5, v_7\},$   
 $\{v_4, v_6, v_7\}, \{v_5, v_6, v_7\},$   
 $\{v_6, v_7, v_8\}, \{v_8, v_9, v_{10}\}$

## Communities:

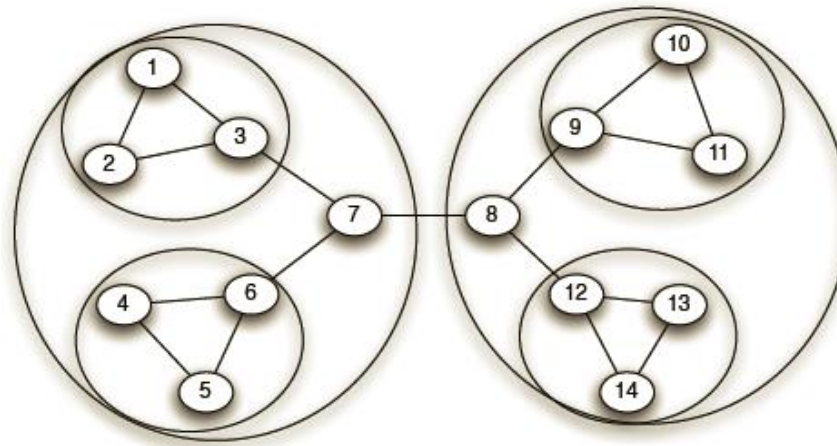
$\{v_1, v_2, v_3\},$   
 $\{v_8, v_9, v_{10}\},$   
 $\{v_3, v_4, v_5, v_6, v_7, v_8\}$



(b) CPM Clique Graph

# Hierarchical Communities

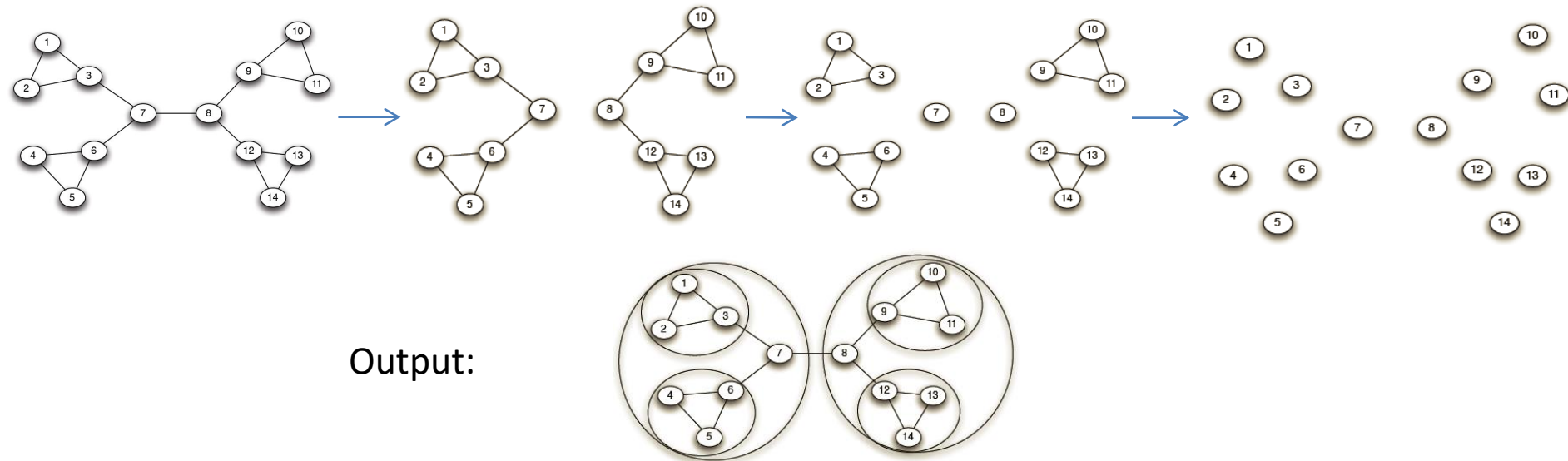
- ❖ Previous methods consider communities at a **single level**
  - Communities may have hierarchies.
    - Each community can have **sub/super communities**.
  - Hierarchical clustering deals with this scenario and generates community hierarchies.



# Girvan-Newman Algorithm

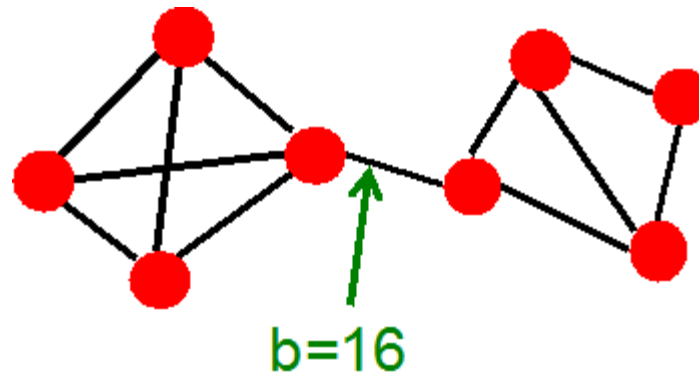
## ❖ Girvan-Newman Algorithm

- Initially  $n$  members are considered as 1 community in hierarchical clustering.
- Split communities by removing links that are important
  - An edge is important if it is a bridge
  - Remove that edge will disconnect the network into connected components
- Repeat until there are no edges

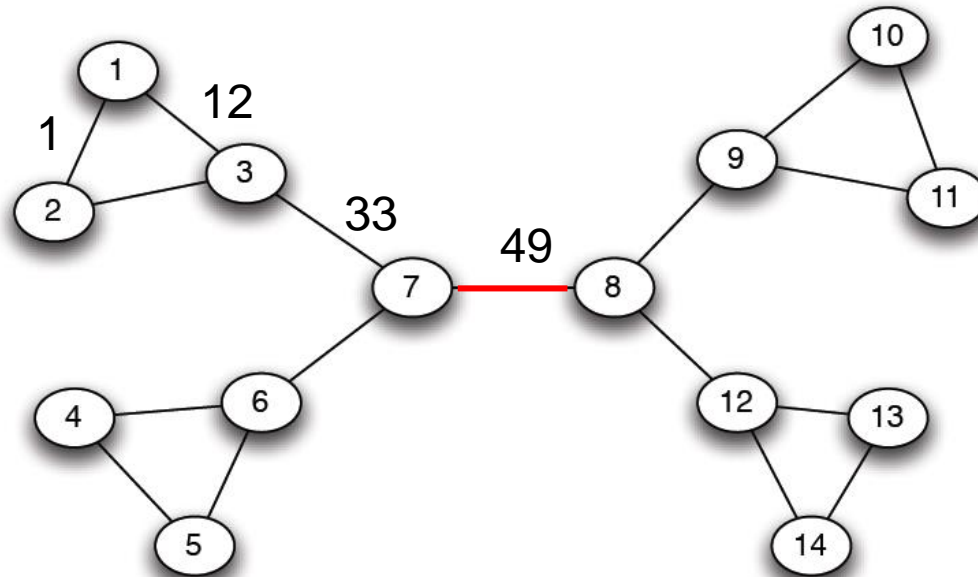


# Girvan-Newman Algorithm: Principle

- ❖ Remove the edges with highest betweenness value to form sub communities
- ❖ Edge betweenness: number of shortest paths between pairs of nodes that run along that edge



# Girvan-Newman algorithm example



$\text{Betweenness}(7,8) = 7 \times 7 = 49$

$\text{Betweenness}(1,3) = 1 \times 12 = 12$

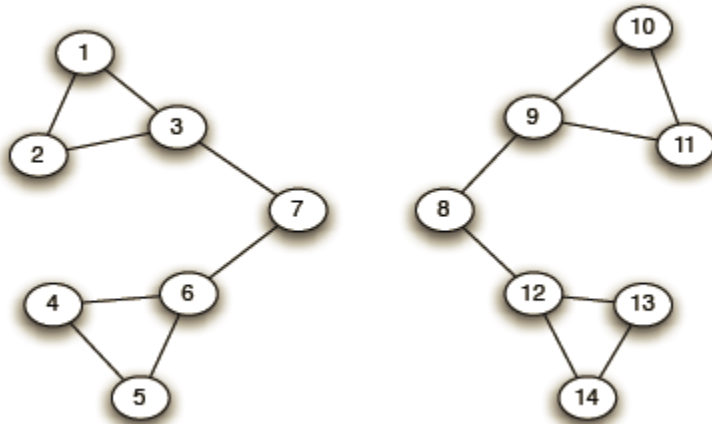
$\text{Betweenness}(3,7) = \text{betweenness}(6,7) = \text{betweenness}(8,9) = \text{betweenness}(8,12) = 3 \times 11 = 33$



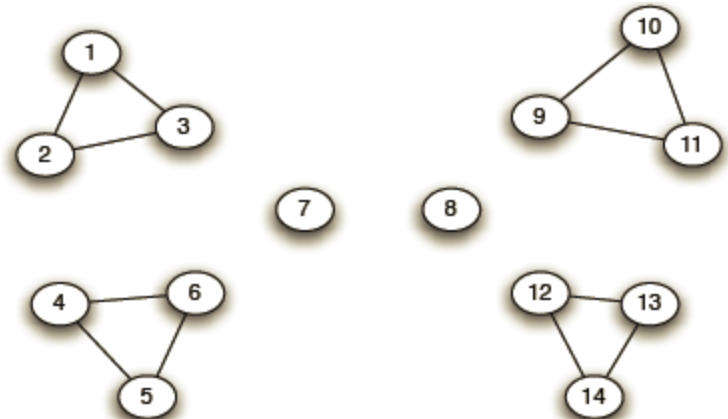
# Example

Need to re-compute betweenness at every step

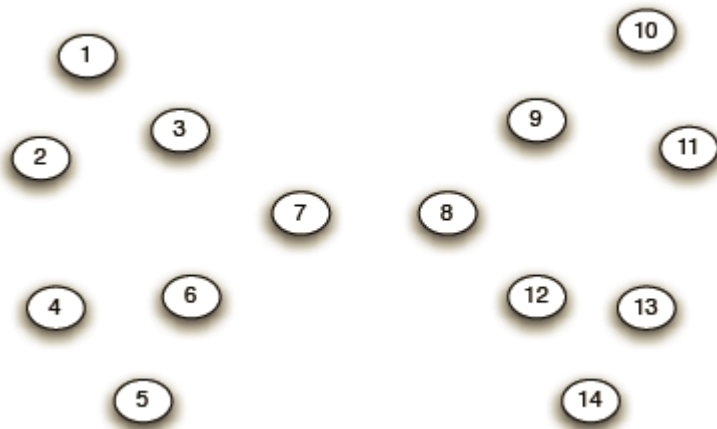
**Step 1**



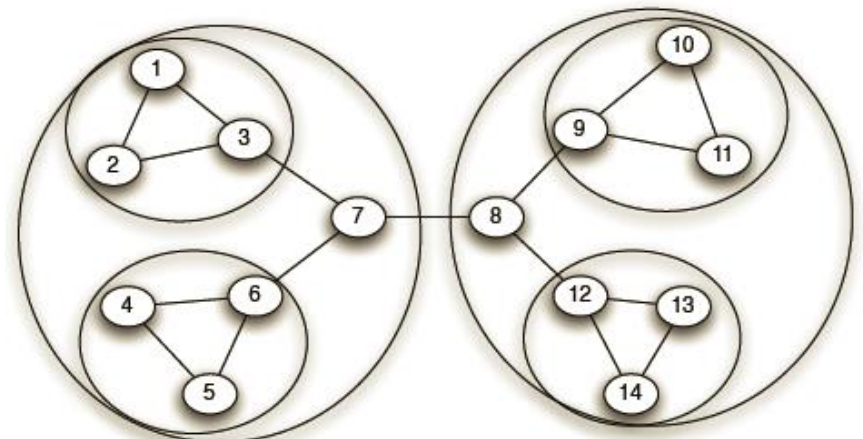
**Step 2**



**Step 3**



**Hierarchical network decomposition**



# Louvain algorithm (OPTIONAL)

- ❖ Procedure:
  - Construct small communities by optimizing the modularity of the network
  - Group small communities into one new community node
  - Repeat until no new communities are formed
- ❖ Output: hierarchical structure with each level corresponding to each grouping step

# Modularity – Community Quality

- ❖ We define now the modularity measure  $Q$ 
  - $A_{ij}$  = effective number of edges between nodes  $i$  and  $j$
  - $c_i, c_j$  = communities of nodes  $i$  and  $j$
  - $m$  = total number of edges
  - $k_i$  = number of outgoing edges of node  $i$  (degree)

Effective number of edges

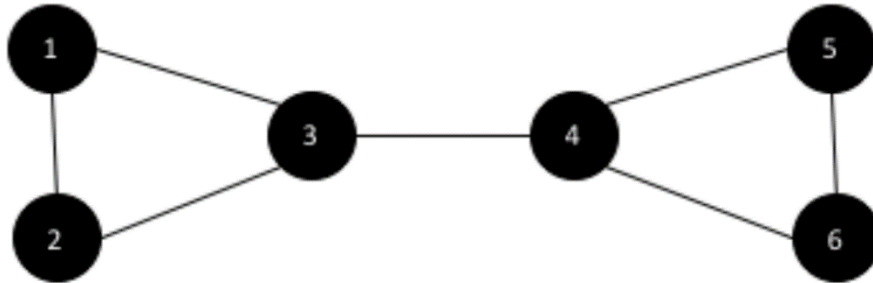
Expected number of edges

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

- ❖ Properties
  - $Q$  in  $[-1,1]$
  - $0.3-0.7 < Q$  means significant community structure

# Louvain algorithm example

- ❖ Initial modularity:  $Q=0$
- ❖ Start processing nodes in order



How to compute modularity? See [6]

# Louvain algorithm example

## Processing Node 1

### 1. If joining node 1 to node 2:

➤ Old modularity:  $Q = 0$

➤ New modularity:  $Q = \frac{1}{2 \times 7} \times \left(1 - \frac{2 \times 2}{2 \times 7}\right) = \frac{1}{14} \times \frac{10}{14} > 0$

➤  $\Delta Q = \frac{1}{14} \times \frac{10}{14} (*)$

### 2. If joining node 1 to node 3:

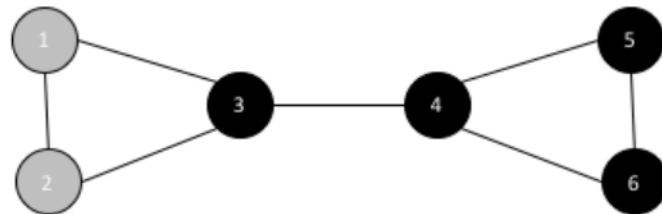
➤ Old modularity:  $Q = 0$

➤ New modularity:  $Q = \frac{1}{2 \times 7} \times \left(1 - \frac{2 \times 3}{2 \times 7}\right) = \frac{1}{14} \times \frac{8}{14} > 0$

➤  $\Delta Q = \frac{1}{14} \times \frac{8}{14} (**)$

→ joining node 1 to node 2 is better ( $*$  >  $**$ )

New modularity:  $Q = \frac{1}{14} \times \frac{10}{14}$



# Louvain algorithm example

## ❖ Processing Node 2:

1. Joining node 2 to node 3 (leaving community of node {1,2})

- Old modularity:  $Q = \frac{1}{14} \times \frac{10}{14}$
- New modularity:  $Q = \frac{1}{2 \times 7} \times \left(1 - \frac{2 \times 3}{2 \times 7}\right) = \frac{1}{14} \times \frac{8}{14}$

→ No improvement

## ❖ Processing Node 3

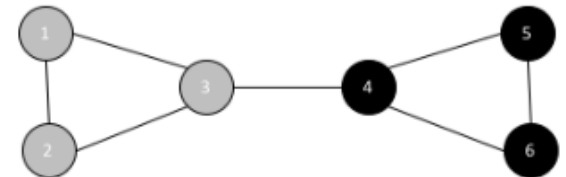
1. Joining node 3 to community {1,2} (via node 1 or 2):

- Old modularity:  $Q = \frac{1}{14} \times \frac{10}{14}$
- New modularity  $Q = \frac{1}{2 \times 7} \times \left(3 - \frac{2 \times 2}{2 \times 7} - \frac{2 \times 3}{2 \times 7} - \frac{2 \times 3}{2 \times 7}\right) = \frac{1}{14} \times \frac{26}{14}$

2. Joining node 3 to node 4:

- Old modularity:  $Q = \frac{1}{14} \times \frac{10}{14}$
- New modularity  $Q = \frac{1}{14} \times \frac{10}{14} + \frac{1}{2 \times 7} \times \left(1 - \frac{3 \times 3}{2 \times 7}\right) = \frac{1}{14} \times \frac{15}{14}$

➤ Option 1 is the best (highest modularity gain)



# Louvain algorithm example

## ❖ Processing Node 4

1. Joining node 4 with community {1,2,3} via 3

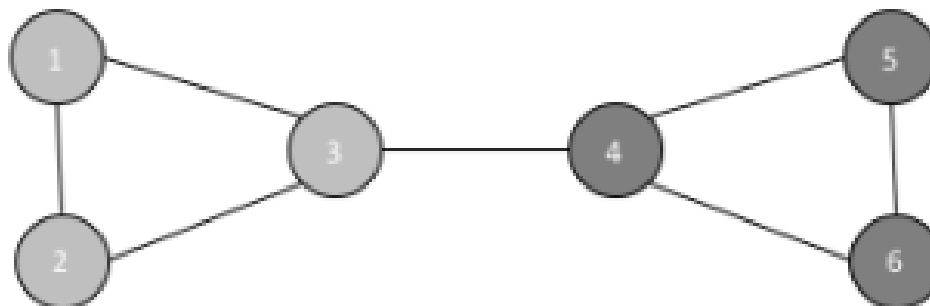
$$\blacksquare Q = \frac{1}{14} \times \left( 4 - \frac{2 \times 2}{14} - \frac{2 \times 3}{14} - \frac{2 \times 3}{14} - \frac{3 \times 3}{14} - \frac{2 \times 3}{14} - \frac{2 \times 3}{14} \right) = \frac{1}{14} \times \frac{19}{14}$$

2. Joining node 4 with node 5

$$\blacksquare Q = \frac{1}{14} \times \frac{26}{14} + \frac{1}{2 \times 7} \times \left( 1 - \frac{2 \times 3}{2 \times 7} \right) = \frac{1}{14} \times \frac{34}{14}$$

→ option 2 is better

❖ Finally, nodes 4,5,6 will join the second community



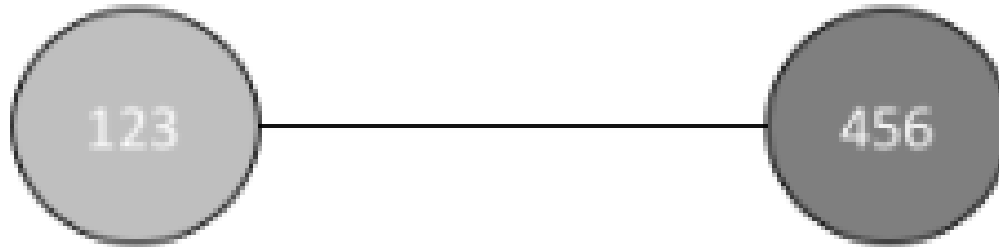
# Louvain algorithm example

Repeat the procedure on the new merged nodes:

❖  $m=1$

❖  $Q = \frac{1}{2 \times 1} \left( 1 - \frac{1 \times 1}{2} \right) > 0$

→ {1,2,3} will merge with {4,5,6}





# IV. Information Diffusion

- ❖ **Information diffusion:** process by which a piece of information (knowledge) is spread and reaches individuals through interactions.
- ❖ Studied in a plethora of sciences.
- ❖ We discuss methods from
  - Sociology, epidemiology, and ethnography
  - All are useful for social media mining.

<https://www.youtube.com/watch?v=9QnfWhtujPA>

# Notable example

- ❖ Between 1996/1997,
  - Hotmail was one of the first internet businesses to become extremely successful utilizing viral marketing
  - By inserting the tagline “*Get your free e-mail at Hotmail*” at the bottom of every e-mail sent out by its users.
- ❖ Hotmail was able to sign up **12 million users** in 18 months.
- ❖ By the time Hotmail reached **66 million** users, the company was establishing **270,000** new accounts each day.



# Information Diffusion Process

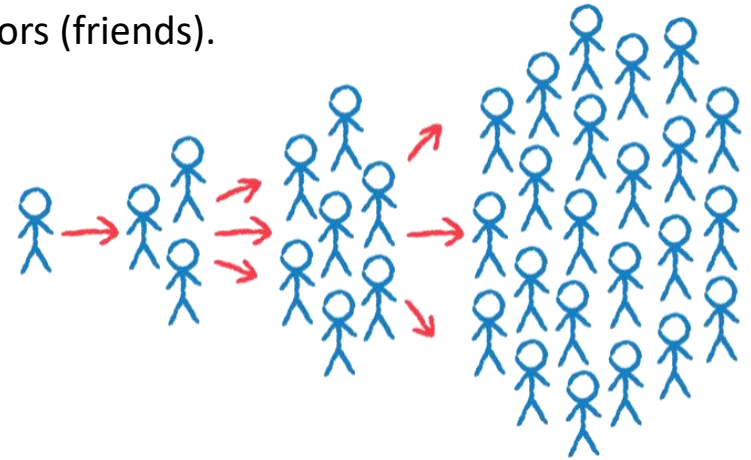
- ❖ **Sender(s).** A sender or a small set of senders that initiate the information diffusion process;
- ❖ **Receiver(s).** A receiver or a set of receivers that receive diffused information. Commonly, the set of receivers is much larger than the set of senders and can overlap with the set of senders; and
- ❖ **Medium.** This is the medium through which the diffusion takes place. For example, when a rumor is spreading, the medium can be the personal communication between individuals

# Information Diffusion in Social Media

- ❖ Users often repost content posted by others in the network.
  - Content is often received via immediate neighbors (friends).



Information propagates  
through friends



- An information cascade:** a piece of information/decision cascaded among some users, where
- individuals are connected by a network and
  - individuals are only observing decisions of their immediate neighbors (friends).

# Information Diffusion Model

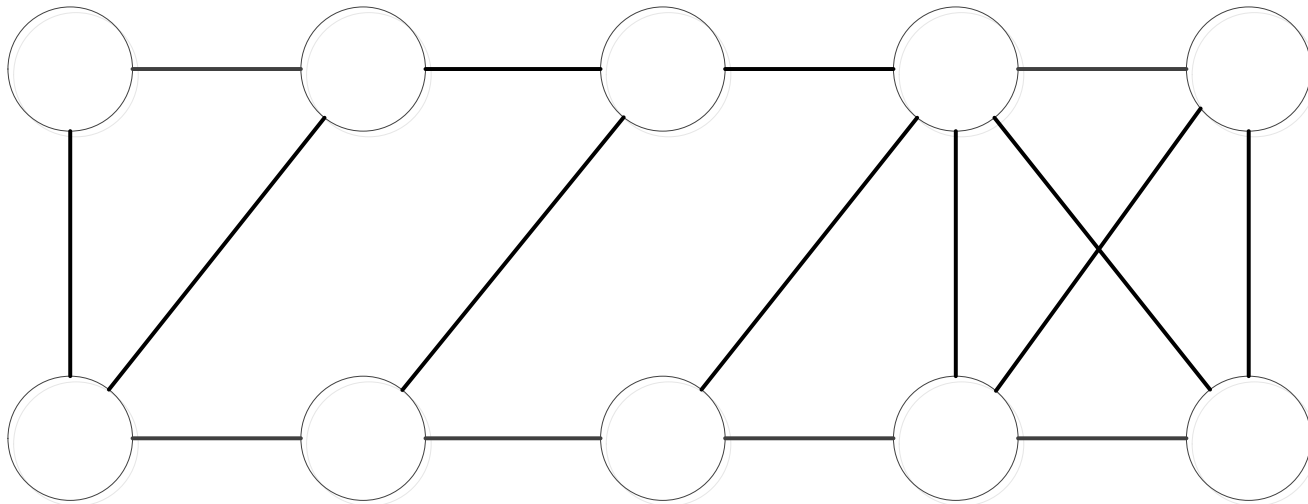
- ❖ The network is a directed graph.
  - Nodes are actors
  - Edges depict the communication channels between them.
- ❖ Nodes can be:
  - **Active:** the node has adopted the behavior/innovation/decision/idea;
  - **Inactive:** not active
- ❖ An activated node can activate its neighboring nodes; and
- ❖ Activation is a progressive process, where nodes change from inactive to active, but not vice versa

# Influence Maximization

- ❖ The problem of finding a small set of nodes in a social network such that their aggregated spread in the network is maximized
- ❖ **Given**
  - A limited budget **B** for initial advertising
    - Example: give away free samples of product
  - Estimated spread between individuals
- ❖ **Goal**
  - To trigger a large spread
    - i.e., further adoptions of a product
- ❖ **Question**
  - Which set of individuals should be targeted at the very beginning?

# Influence Maximization: Example

- ❖ You are allowed to select  $k$  seeds
- ❖ A node is activated if  $\geq 1/2$  of its neighbors is activate



# Maximizing the Spread of Cascade

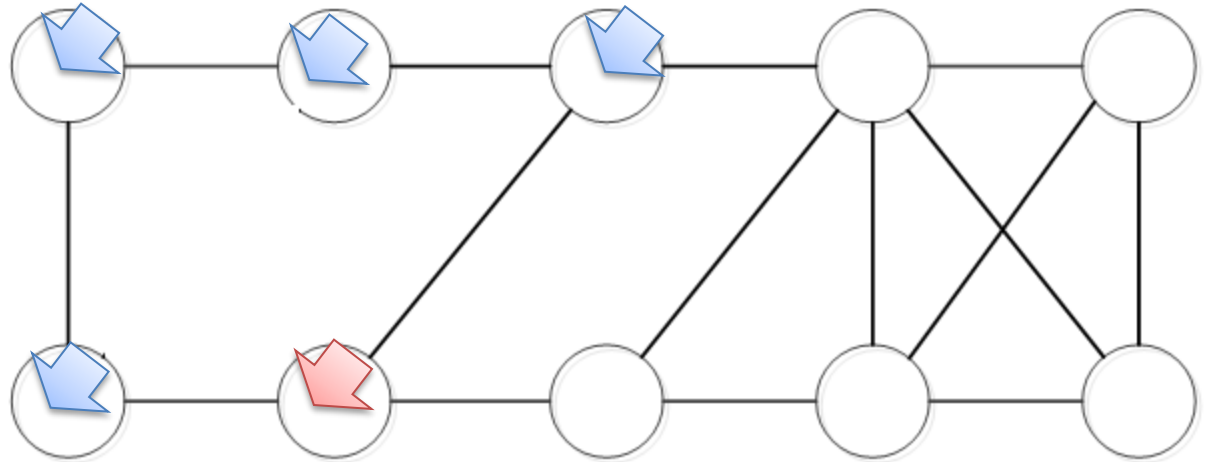


seed

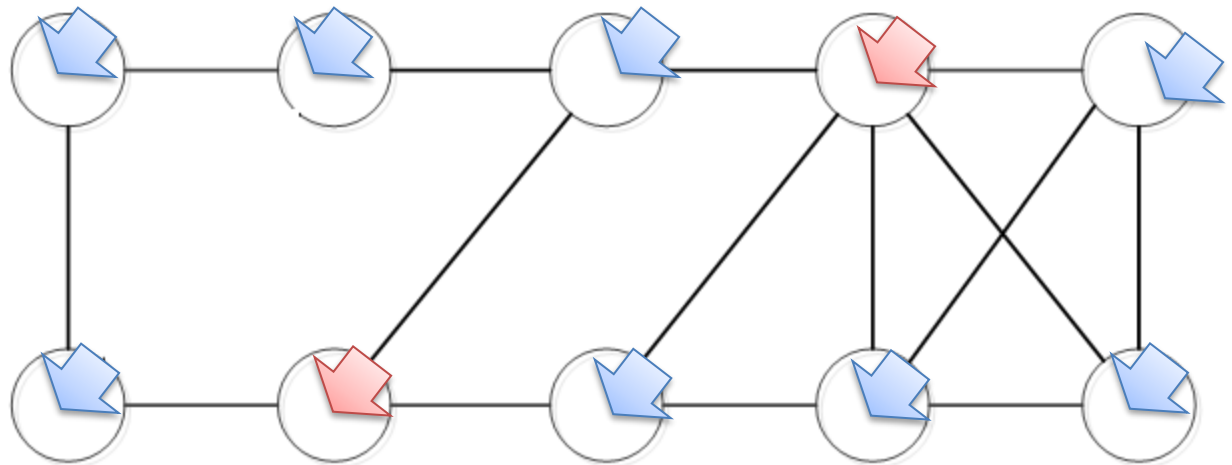


activated nodes

**k=1 seed**



**k=2 seeds**





# Problem Statement

- ❖ **Influence** of node set  $S$ :  $f(S)$ 
  - An **expected number** of activated nodes at the end of the cascade, if set  $S$  is the initial active set
- ❖ **Problem:**
  - Given a parameter  $k$  (budget), find a  $k$ -node set  $S$  to maximize  $f(S)$
  - A constrained optimization problem with  $f(S)$  as the objective function
- ❖ It is NP-hard to determine the optimum set for influence maximization

# Greedy Algorithm

❖ We can use a **greedy** algorithm

- Start with an empty set  $S$
- For  $k$  iterations:
  - Add node  $v$  to  $S$  that maximizes  $f(S \cup \{v\}) - f(S)$ .

❖ **Guarantee:**

- Theorem: the greedy algorithm provides a  **$(1 - 1/e)$  approximation**.
- I.e. The resulting set  $S$  activates **at least**  $(1 - 1/e) \approx 63\%$  of the number of nodes that any size- $k$  set  $S$  could activate

---

**Algorithm 1** Maximizing the spread of cascades – Greedy algorithm

---

**Require:** Diffusion graph  $G(V, E)$ , budget  $k$

```
1: return Seed set  $S$  (set of initially activated nodes)
2:  $i = 0$ ;
3:  $S = \{\}$ ;
4: while  $i \neq k$  do
5:    $v = \arg \max_{v \in V \setminus S} f(S \cup \{v\})$ ;
   or equivalently  $\arg \max_{v \in V \setminus S} f(S \cup \{v\}) - f(S)$ 
6:    $S = S \cup \{v\}$ ;
7:    $i = i + 1$ ;
8: end while
9: Return  $S$ ;
```

---

# Summary

## I. Network Representation

- Directed or undirected graph

## II. Centrality Analysis

- Different types of centrality: degree, eigenvector, Pagerank

## III. Community Analysis

- Clique Percolation Method: produces overlapping and single-level communities
- Girvan-Newman algorithm: produces disjoint and hierarchical communities

## IV. Information Diffusion Analysis

- A node can be influenced by its neighbors
- Influence maximization is NP-hard

# References

- [1] <https://youtu.be/Qj2uWpYsdCM>
- [2] R. Zafarani, M. A. Abbasi, and H. Liu, Social Media Mining: An Introduction, Cambridge University Press, 2014. Free book and slides at <http://socialmediamining.info/>
- [3] Representation Learning on Networks, [snap.stanford.edu/proj/embeddings-www](http://snap.stanford.edu/proj/embeddings-www), WWW 2018
- [4] <http://sociograph.blogspot.ch/2011/11/clique-percolation-in-few-lines-of.html>
- [5] <https://friendsarena.wordpress.com/category/uncategorized/>