Social Media Mining

Graph Essentials
Graph Basics
Nodes and Edges

A network is a graph

- **nodes, actors, or vertices** (plural of vertex)
- Connections, **edges** or **ties**

![Graph Diagram]

- **Nodes** (v1, v2, v3, v4, v5, v6, v7)
- **Edges** (connections between nodes)

**Edge**

**Node**
Nodes and Edges

- In a social graph, nodes are people and any pair of people connected denotes the friendship, relationships, social ties between them.
- In a web graph, “nodes” represent sites and the connection between nodes indicates web-links between them.

- The size of the graph is $|V| = n$
- Number of edges (size of the edge-set) $|E| = m$

$$V = \{v_1, v_2, \ldots, v_n\}$$
$$E = \{e_1, e_2, \ldots, e_m\}$$
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 \( e(v_2, v_1) \). In undirected graphs both representations are the same
Neighborhood and Degree (In-degree, out-degree)

• For any node $v$, the set of nodes it is connected to via an edge is called its **neighborhood** and is represented as $N(v)$

• The number of edges connected to one node is the degree of that node (the size of its neighborhood)
  – Degree of a node $i$ is usually presented using notation $d_i$
  – In case of directed graphs
    \[ d_i^{\text{in}} \] • In-degrees is the number of edges pointing towards a node
    \[ d_i^{\text{out}} \] • Out-degree is the number of edges pointing away from a node
Degree and Degree Distribution

- **Theorem 1.** The summation of degrees in an undirected graph is twice the number of edges

\[ \sum_i d_i = 2|E| \]

- **Lemma 1.** The number of nodes with odd degree is even

- **Lemma 2.** In any directed graph, the summation of in-degrees is equal to the summation of out-degrees,

\[ \sum_i d_{i}^{\text{out}} = \sum_j d_{j}^{\text{in}} \]
Degree Distribution

When dealing with very large graphs, how nodes’ degrees are distributed is an important concept to analyze and is called **Degree Distribution** $p_d$.

$$p_d = \frac{n_d}{n};$$

- Where $n_d$ is the number of nodes with degree $d$.
- Degree distribution can be computed from degree sequence:

$$\pi(d) = \{d_1, d_2, \ldots, d_n\}$$

**Degree distribution histogram**

- The x-axis represents the degree and the y-axis represents the number of nodes (frequency) having that degree.

![Degree Distribution Graph](image)
Subgraph

- Graph G can be represented as a pair G(V, E), where V is the node set and E is the edge set.

- G'(V', E') is a subgraph of G(V, E) (induced subgraph)

\[ V' \subseteq V, \]
\[ E' \subseteq (V' \times V') \cap E \]
Graph Representation

• Adjacency Matrix
• Adjacency List
• Edge List
Adjacency Matrix

\[ A_{ij} = \begin{cases} 
1, & \text{if there is an edge between nodes } v_i \text{ and } v_j \\
0, & \text{otherwise} 
\end{cases} \]

Diagonal Entries are self-links or loops

Social media networks have very sparse adjacency matrices
Adjacency List

- In an adjacency list for every node, we maintain a list of all the nodes that it is connected to.
- The list is usually sorted based on the node order or other preferences.

<table>
<thead>
<tr>
<th>Node</th>
<th>Connected To</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>$v_2$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$v_1, v_3, v_4$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$v_2, v_4$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>$v_2, v_3, v_5, v_6$</td>
</tr>
<tr>
<td>$v_5$</td>
<td>$v_4$</td>
</tr>
<tr>
<td>$v_6$</td>
<td>$v_4$</td>
</tr>
</tbody>
</table>
Edge List

• In this representation, each element is an edge and is usually represented as \((u, v)\), denoting that node \(u\) is connected to node \(v\) via an edge.
Types of Graphs

• Null, Empty, Directed/Undirected/Mixed, Simple/Multigraph, Weighted, Signed Graph
• The adjacency matrix for directed graphs is not symmetric \((A \neq A^T)\)
  
  \(-\ (A_{ij} \neq A_{ji})\)

• The adjacency matrix for undirected graphs is symmetric \((A = A^T)\)
Simple Graphs and Multigraphs

- Simple graphs are graphs where only a single edge can be between any pair of nodes.
- Multigraphs are graphs where you can have *multiple edges* between two nodes and loops.

![Simple graph](image)

![Multigraph](image)

- The adjacency matrix for multigraphs can include numbers larger than one, indicating multiple edges between nodes.
Weighted Graph

• A weighted graph is one where edges are associated with weights
  – For example, a graph could represent a map where nodes are cities and edges are routes between them
  • The weight associated with each edge could represent the distance between these cities

\[ G(V, E, W) \]

\[ A_{ij} = \begin{cases} 
  w, & w \in \mathbb{R} \\
  0, & \text{There is no edge between } i \text{ and } j 
\end{cases} \]
Signed Graph

- When weights are binary (0/1, -1/1, +/-) we have a **signed** graph

- It is used to represent **friends** or **foes**
- It is also used to represent **social status**
Connectivity in Graphs

• Adjacent nodes/Edges, Walk/Path/Trail/Tour/Cycle,
Adjacent nodes and Incident Edges

Two nodes are adjacent if they are connected via an edge.

Two edges are incident, if they share on end-point.

When the graph is directed, edge directions must match for edges to be incident.
Walk: A walk is a sequence of incident edges visited one after another

- **Open walk**: A walk does not end where it starts
- **Close walk**: A walk returns to where it starts

- Representing a walk:
  - A sequence of edges: $e_1, e_2, ..., e_n$
  - A sequence of nodes: $v_1, v_2, ..., v_n$

- Length of walk: the number of visited edges

Length of walk = 8
Path

- A walk where **nodes and edges are distinct** is called a **path** and a closed path is called a **cycle**
- The **length of a path** or cycle is the number of edges visited in the path or cycle

Length of path = 4
Random walk

- A walk that in each step the next node is selected *randomly among the neighbors*
  - The weight of an edge can be used to define the probability of visiting it
  - For all edges that start at $v_i$ the following equation holds

\[
\sum_x w_{i,x} = 1, \forall i, jw_{i,j} \geq 0.
\]
Connectivity

- A node $v_i$ is **connected** to node $v_j$ (or reachable from $v_j$) if it is adjacent to it or there exists a path from $v_i$ to $v_j$.

- A graph is **connected**, if there exists a path between any pair of nodes in it
  - In a directed graph, a graph is **strongly connected** if there exists a directed path between any pair of nodes
  - In a directed graph, a graph is **weakly connected** if there exists a path between any pair of nodes, without following the edge directions

- A graph is **disconnected**, if it not connected.
Connectivity: Example

(a) Connected  (b) Disconnected  (c) Strongly connected  (d) Weakly connected
Component

• A **component** in an undirected graph is a **connected subgraph**, i.e., there is a path between every pair of nodes inside the component.

• In directed graphs, we have a **strongly connected components** when there is a path from \( u \) to \( v \) and one from \( v \) to \( u \) for every pair \( (u,v) \).

• The component is **weakly connected** if replacing directed edges with undirected edges results in a connected component.
Component Examples:

3 components

3 Strongly-connected components
Shortest Path

- **Shortest Path** is the path between two nodes that has the shortest length.

- The concept of the neighborhood of a node can be generalized using shortest paths. An **n-hop neighborhood** of a node is the set of nodes that are within n hops distance from the node.
Diameter

- The **diameter** of a graph is the length of the longest shortest path between any pair of nodes between any pairs of nodes in the graph.

\[
diameter_G = \max_{(v_i,v_j) \in V \times V} l_{i,j}.
\]

- How big is the diameter of the web?
Special Graphs
Trees and Forests

- **Trees** are special cases of undirected graphs
- A tree is a graph structure that has no cycle in it
- In a tree, there is exactly one path between any pair of nodes
- In a tree: $|V| = |E| + 1$

- A set of disconnected trees is called a **forest**

A forest containing 3 trees
Special Subgraphs
Spanning Trees

- For any connected graph, the **spanning tree** is a subgraph and a tree that includes all the nodes of the graph.
- There may exist multiple spanning trees for a graph.
- For a weighted graph and one of its spanning tree, the weight of that spanning tree is the summation of the edge weights in the tree.
- Among the many spanning trees found for a weighted graph, the one with the minimum weight is called the **minimum spanning tree (MST)**.
Steiner Trees

• Given a weighted graph $G : (V, E, W)$ and a subset of nodes $V' \subseteq V$ (terminal nodes), the Steiner tree problem aims to find a tree such that it spans all the $V'$ nodes and the weight of this tree is minimized.
Complete Graphs

- A complete graph is a graph where for a set of nodes $V$, *all possible edges* exist in the graph.
- In a complete graph, any pair of nodes are connected via an edge.
Planar Graphs

• A graph that can be drawn in such a way that no two edges cross each other (other than the endpoints) is called planar
A bipartite graph $G(V; E)$ is a graph where the node set can be partitioned into two sets such that, for all edges, one end-point is in one set and the other end-point is in the other set.

\[
\begin{align*}
V &= V_L \cup V_R, \\
V_L \cap V_R &= \emptyset, \\
E &\subset V_L \times V_R.
\end{align*}
\]
Affiliation Networks

• An affiliation network is a bipartite graph. If an individual is associated with an affiliation, an edge connects the corresponding nodes.
Regular Graphs

• A regular graph is one in which all nodes have the same degree
• Regular graphs can be connected or disconnected
• In a k-regular graph, all nodes have degree k
• Complete graphs are examples of regular graphs
Bridges (cut-edges)

- Bridges are edges whose removal will increase the number of connected components
Graph Algorithms
Graph/Network Traversal Algorithms
Traversals
1. All users are visited; and
2. No user is visited more than once.

- There are two main techniques:
  - Depth-First Search (DFS)
  - Breadth-First Search (BFS)
Depth-First Search (DFS)

- Depth-First Search (DFS) starts from a node $i$, selects one of its neighbors $j$ from $N(i)$ and performs Depth-First Search on $j$ before visiting other neighbors in $N(i)$.

- The algorithm can be used both for trees and graphs
  - The algorithm can be implemented using a stack structure
Algorithm 2.2 Depth-First Search (DFS)

Require: Initial node $v$, graph/tree $G: (V, E)$, stack $S$

1: return An ordering on how nodes in $G$ are visited
2: Push $v$ into $S$;
3: $visitOrder = 0$;
4: while $S$ not empty do
5: \hspace{1em} node = pop from $S$;
6: \hspace{1em} if node not visited then
7: \hspace{2em} $visitOrder = visitOrder + 1$;
8: \hspace{2em} Mark node as visited with order $visitOrder$; // or print node
9: \hspace{2em} Push all neighbors/children of node into $S$;
10: end if
11: end while
12: Return all nodes with their visit order.
Depth-First Search (DFS): An Example

(1) Depth-First Search (DFS): An Example

(2)

(3)

(4)

(5)

(6)
Breadth-First Search (BFS)

• BFS starts from a node, visits all its immediate neighbors first, and then moves to the second level by traversing their neighbors.

• The algorithm can be used both for trees and graphs
  – The algorithm can be implemented using a queue structure
Algorithm 2.3 Breadth-First Search (BFS)

**Require:** Initial node \( v \), graph/tree \( G(V, E) \), queue \( Q \)

1: \textbf{return} An ordering on how nodes are visited
2: Enqueue \( v \) into queue \( Q \);
3: \( visitOrder = 0; \)
4: \textbf{while} \( Q \) not empty \textbf{do}
5: \hspace{0.5em} \( node = \text{dequeue from } Q; \)
6: \hspace{0.5em} \textbf{if} \( node \) not visited \textbf{then}
7: \hspace{1.5em} \( visitOrder = visitOrder + 1; \)
8: \hspace{1.5em} Mark \( node \) as visited with order \( visitOrder; \)/
9: \hspace{1.5em} \textbf{print} node
10: \hspace{1.5em} Enqueue all neighbors/children of \( node \) into \( Q \);
11: \hspace{0.5em} \textbf{end if}
12: \textbf{end while}
Breadth-First Search (BFS)
Shortest Path

When a graph is connected, there is a chance that multiple paths exist between any pair of nodes

– In many scenarios, we want the shortest path between two nodes in a graph

• Dijkstra’s Algorithm
  – It is designed for weighted graphs with non-negative edges
  – It finds shortest paths that start from a provided node s to all other nodes
  – It finds both shortest paths and their respective lengths
Dijkstra’s Algorithm: Finding the shortest path

1. **Initiation:**
   - Assign zero to the source node and infinity to all other nodes
   - Mark all nodes unvisited
   - Set the source node as current

2. **For the current node, consider all of its unvisited neighbors and calculate their tentative distances**
   - If tentative distance (current node’s distance + edge weight) is smaller than neighbor’s distance, then Neighbor’s distance = tentative distance

3. **After considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set**
   - A visited node will never be checked again and its distance recorded now is final and minimal

4. **If the destination node has been marked visited or if the smallest tentative distance among the nodes in the unvisited set is infinity, then stop**

5. **Set the unvisited node marked with the smallest tentative distance as the next "current node" and go to step 2**
Dijkstra’s Algorithm Execution Example
Dijkstra’s Algorithm

• Dijkstra’s algorithm is source-dependent and finds the shortest paths between the source node and all other nodes. To generate all-pair shortest paths, one can run dijkstra’s algorithm \( n \) times or use other algorithms such as Floyd-Warshall algorithm.

• If we want to compute the shortest path from source \( v \) to destination \( d \), we can stop the algorithm once the shortest path to the destination node has been determined.
Other slides
Internet
Phoenix Road Network

Social Media Mining

Graph Essentials
Social Networks and Social Network Analysis

• A social network
  – A network where elements have a social structure
    • A set of actors (such as individuals or organizations)
    • A set of ties (connections between individuals)

• Social networks examples:
  – your family network, your friend network, your colleagues, etc.

• To analyze these networks we can use Social Network Analysis (SNA)

• Social Network Analysis is an interdisciplinary field from social sciences, statistics, graph theory, complex networks, and now computer science
Social Networks: Examples

High school friendship

High school dating
Webgraph

- A webgraph is a way of representing how internet sites are connected on the web.
- In general, a web graph is a directed multigraph.
- Nodes represent sites and edges represent links between sites.
- Two sites can have multiple links pointing to each other and can have loops (links pointing to themselves).
Webgraph: Government Agencies
Prim’s Algorithm: Finding Minimum Spanning Tree

• It finds minimal spanning trees in a weighted graph
  – It starts by selecting a random node and adding it to the spanning tree.
  – It then grows the spanning tree by selecting edges which have one endpoint in the existing spanning tree and one endpoint among the nodes that are not selected yet. Among the possible edges, the one with the minimum weight is added to the set (along with its end-point).
  – This process is iterated until the graph is fully spanned
Prim’s Algorithm Execution Example
Algorithm 2.7 Bridge Detection Algorithm

Require: Connected graph $G(V, E)$
1: return Bridge Edges
2: $bridgeSet = \emptyset$
3: for $e(u, v) \in E$ do
4: $G' = \text{Remove } e \text{ from } G$
5: $\text{Disconnected} = \text{False}$;
6: if BFS in $G'$ starting at $u$ does not visit $v$ then
7: $\text{Disconnected} = \text{True}$;
8: end if
9: if $\text{Disconnected}$ then
10: $bridgeSet = bridgeSet \cup \{e\}$
11: end if
12: end for
13: Return $bridgeSet$