Link Prediction

Prof. O-Joun Lee

Dept. of Artificial Intelligence,
The Catholic University of Korea
ojlee@catholic.ac.kr

네트워크 과학 연구실 NETWORK SCIENCE LAB

가톨릭대학교 THE CATHOLIC UNIVERSITY OF KOREA

Contents

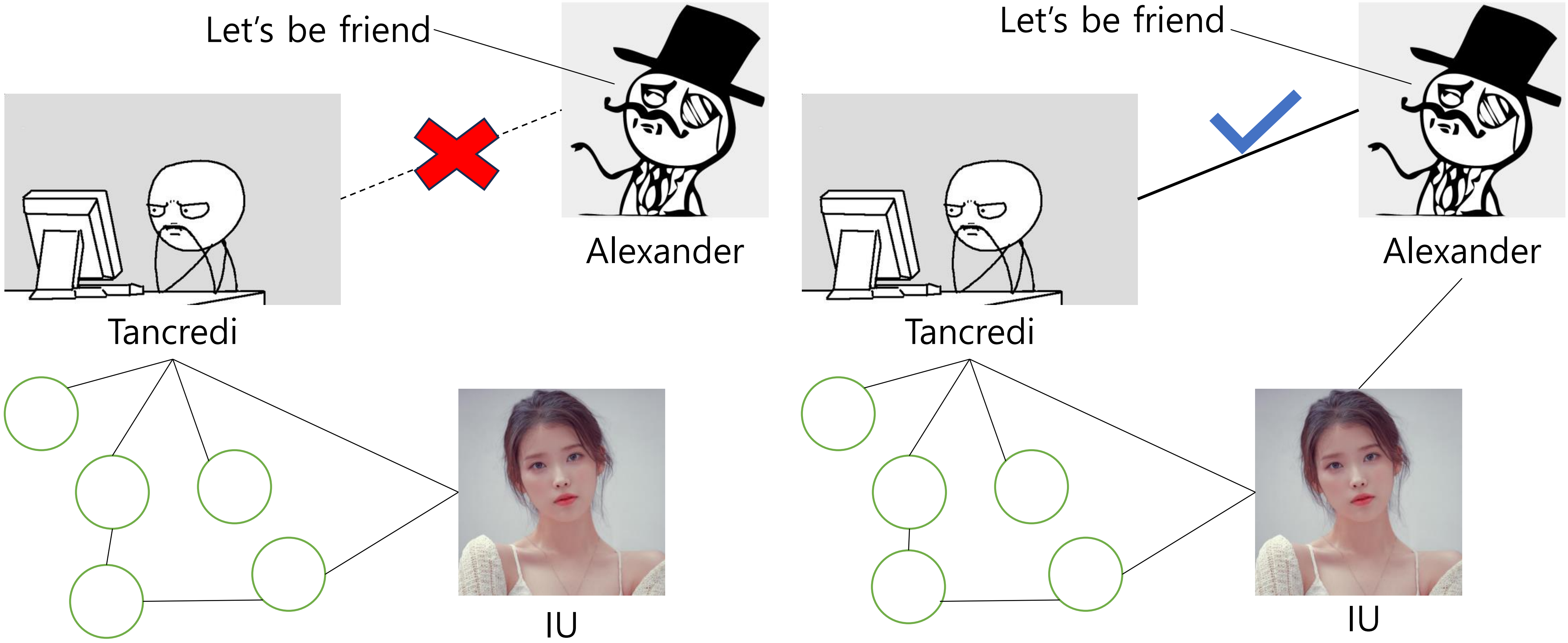
- A primer on Link prediction
- Challenges in Link prediction
- Common methods
  - Common neighbours
  - Jaccard coefficient
  - Adamic-Adar
  - Preferential attachment
  - Path based
  - SimRank
- Evaluation metrics
- Tools for link prediction

Link Prediction in Social Networks

- What we can do in social network prediction:
  - Friend Suggestions
    - Forecast likelihood of connection formation
    - Utilize network structure and historical interactions
  - Community Detection
    - Identify cohesive groups within the network
    - Employ clustering algorithms and network analysis
  - User Behavior Prediction
    - Predict user engagement, churn, or sentiment
    - Inform marketing and product strategies

Link Prediction in Social Networks

- Friend Suggestions task



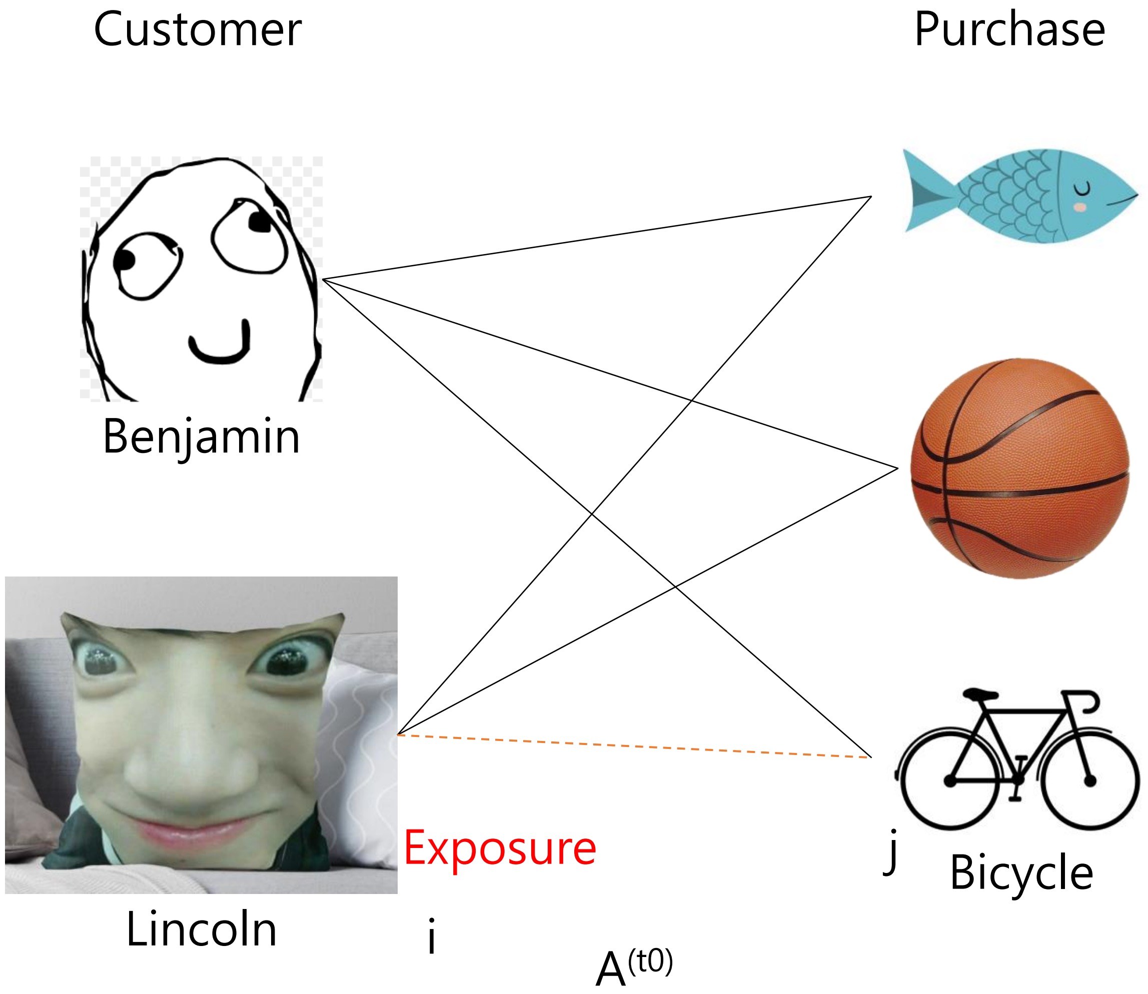
Link Prediction in Recommendation Systems

- What we can do in recommendation systems prediction:
  - Rating Prediction
    - Predict user-item ratings
  - Cold-Start Recommendation
    - Address cold-start problem for new users or items
    - Utilize content-based information or hybrid approaches
  - Implicit Feedback Prediction
    - Predict user interaction likelihood from implicit feedback

Link Prediction in Recommendation Systems

Link prediction as an exposure

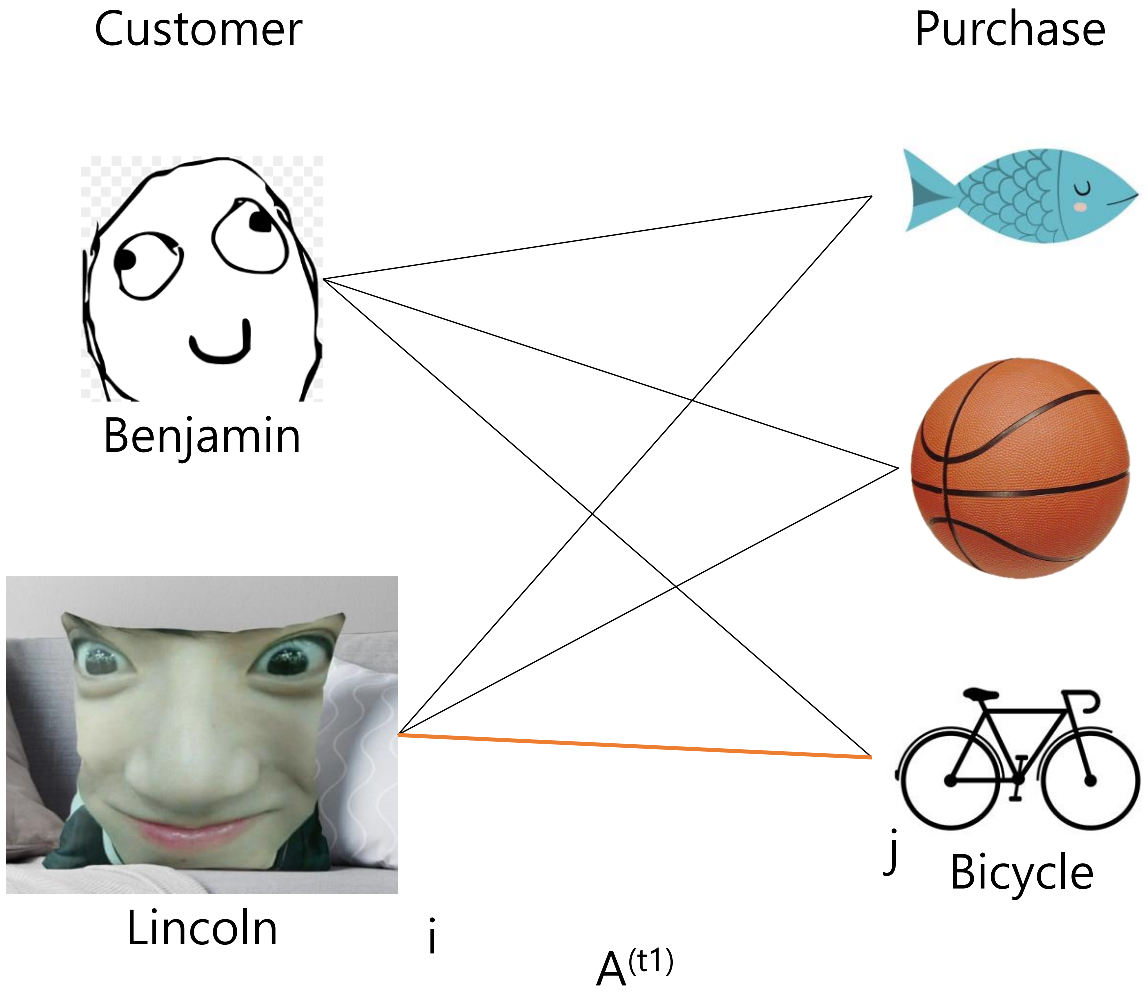
- At time t_0 we expose Lincoln to Bicycle
- We will define this intervention exposure as
  - $E^{(t0)}$ = (Lincoln, Bicycle)



Recommendations as treatments (Joachims et al., 2021)

Link creation outcome

- At time t_1 we see if Lincoln bought Bicycle
- The outcome of the exposure at t_1
  - $A^{(t_1)}_{Lincoln,Bicycle} \in \{0,1\}$



Link Prediction in Chemistry

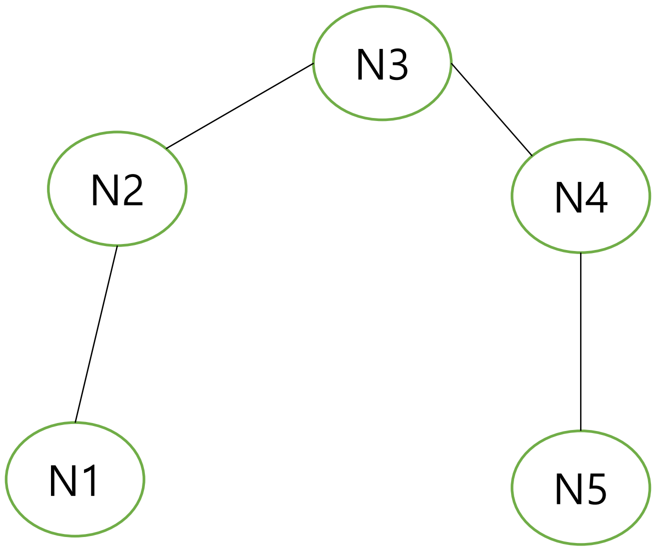
- What we can do in recommendation systems prediction:
  - Chemical Property Prediction
    - Forecast physicochemical properties (e.g., solubility, toxicity)
  - Drug Discovery and Design
    - Identify drug candidates and optimize molecules
  - Reaction Prediction
    - Predict chemical reaction outcomes and pathways

Link Prediction in Transportation Systems

- What we can do in recommendation systems prediction:
  - Traffic Flow Prediction
    - Forecast traffic congestion, flow rates, and travel times
  - Demand Forecasting
    - Predict future demand for transportation services
  - Route Optimization
    - Optimize routes for vehicles, passengers, or freight
  - Accident Prediction
    - Predict the likelihood of traffic accidents and identify high-risk areas
  - Travel Time Estimation
    - Estimate travel times for different transportation modes and routes

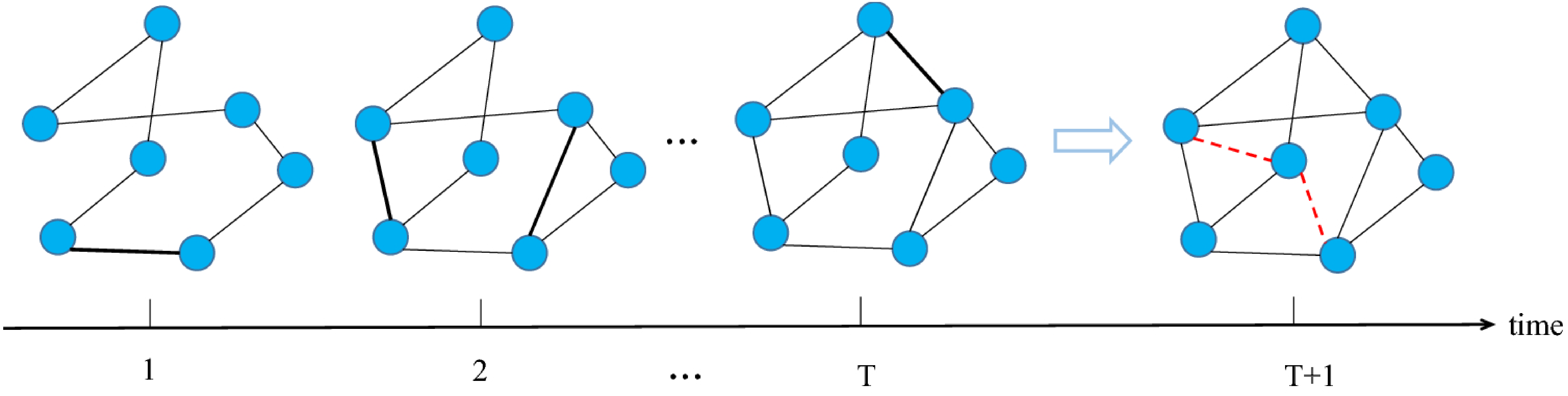
Challenges in Link predictions: Graph sparsity

- Networks often exhibit sparsity, where the number of observed connections is much smaller than the total number of possible connections.
- This makes it challenging to infer missing links accurately, especially in large-scale networks
- A sparse graph has a number of edges closer to the minimal number of edges
- Eg: A graph has 5 nodes and only 4 edges



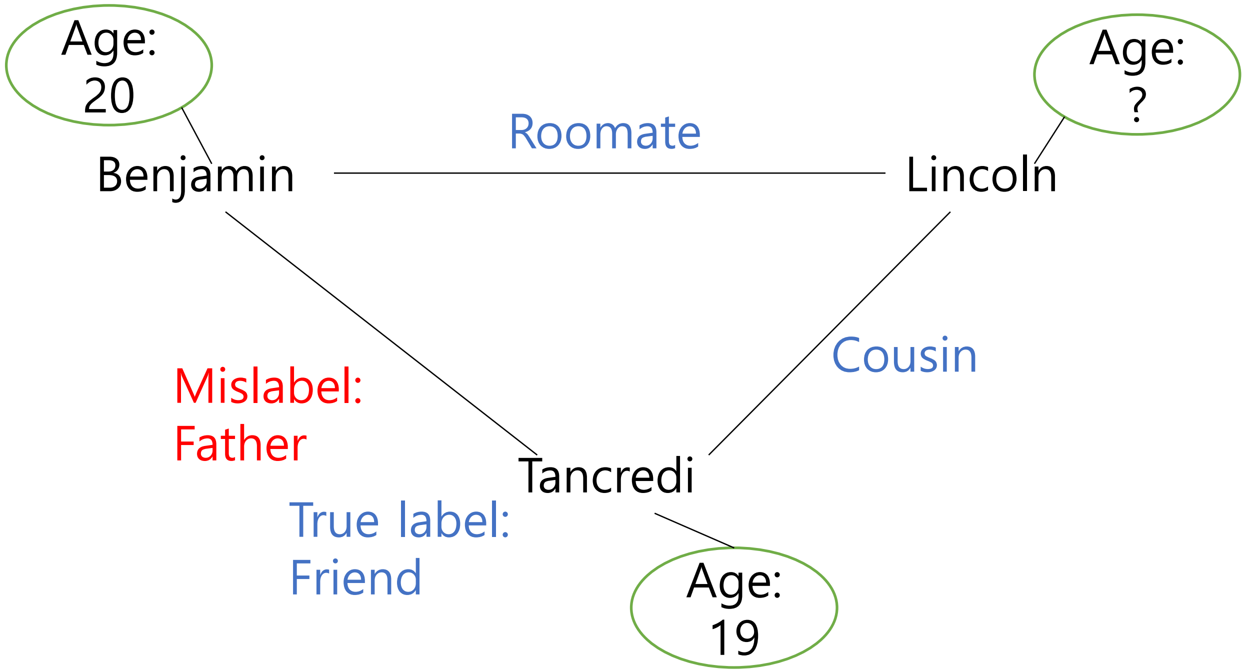
Challenges in Link predictions: Temporal dynamic

- Networks evolve over time, with new connections forming and existing connections breaking.
- Link prediction models must be able to capture and adapt to temporal dynamics, predicting future links based on past network states.



Challenges in Link predictions: Noisy data and missing attributes

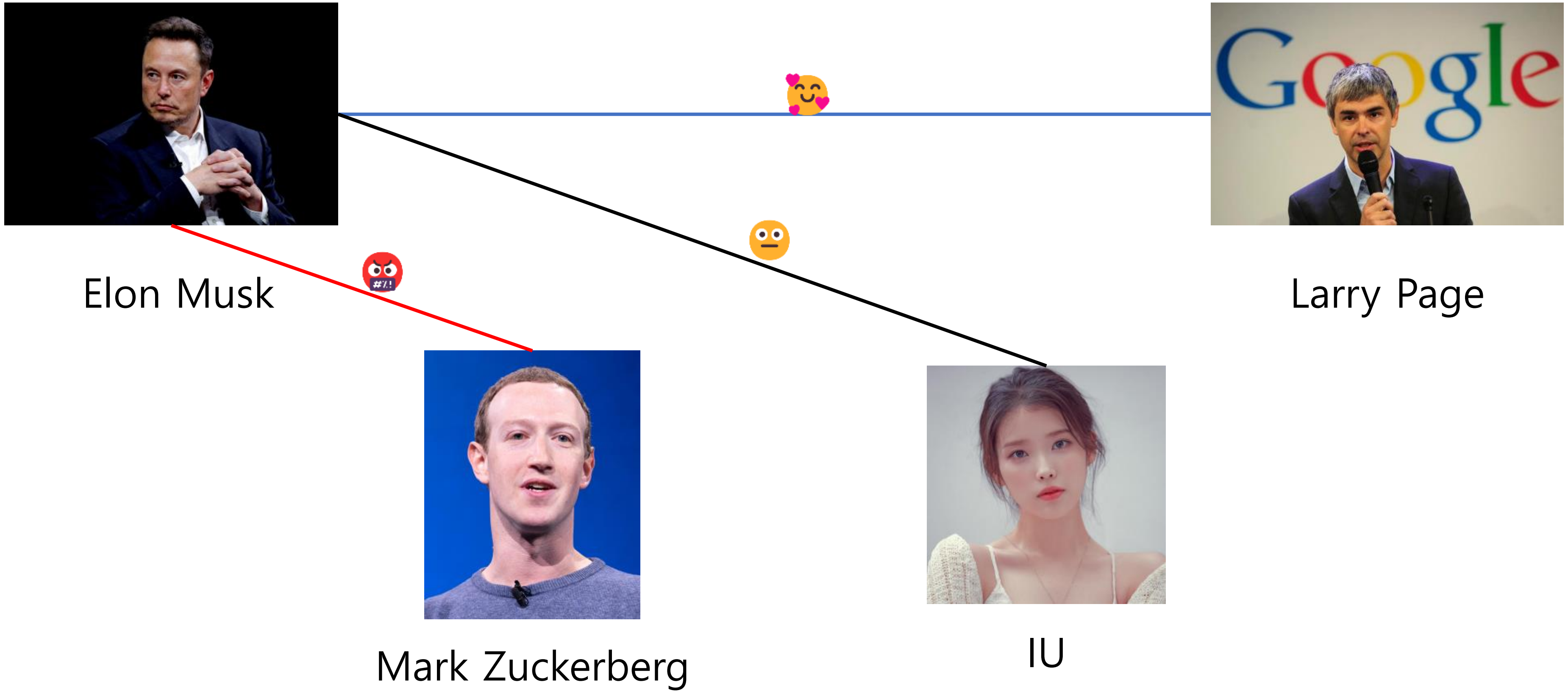
- Link prediction models often rely on node and edge attributes to infer connections.
- However, real-world data may be incomplete or noisy, leading to challenges in feature selection and model training



- There are 3 types of Link Prediction
  - Positive Link
    - Positive links refer to connections or relationships that are expected or desirable in a network
    - These connections typically represent friendships, collaborations, interactions, or any other positive associations between nodes
  - Negative Link
    - Negative links represent relationships or connections that are undesirable or unlikely in a network
    - These connections might indicate conflicts, disagreements, competition, or any other negative associations between nodes
  - Neutral Link
    - Neutral links refer to connections that neither positively nor negatively affect the network
    - These connections are typically neither desirable nor undesirable; they might represent indifference or lack of significant interaction between nodes

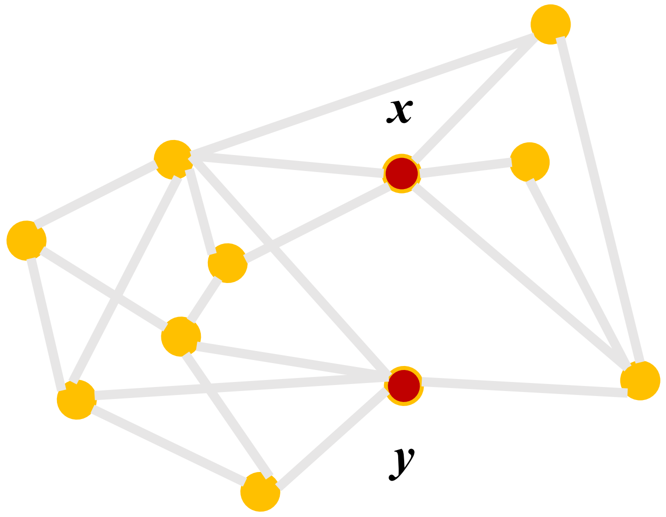
Types of Link Prediction

- There are 3 types of Link Prediction
  - Positive Link
  - Negative Link
  - Neutral Link

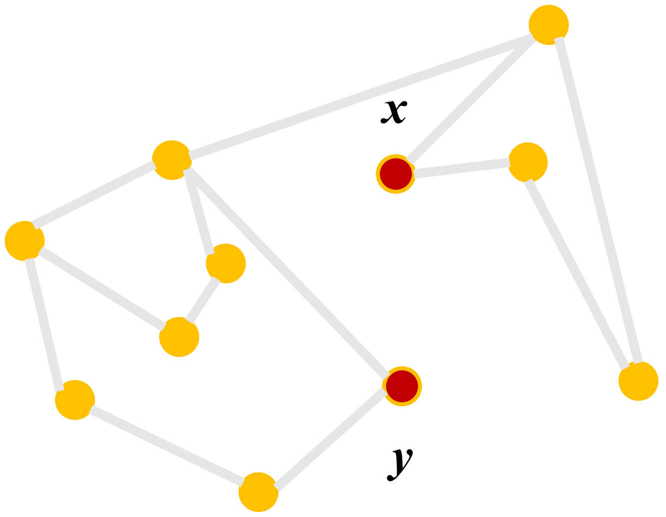


The intuition

- Link prediction also equals to “Relation prediction” or “Graph completion” or “Relational inference” depending on the specific application domain
- In many networks, people who are “close” belong to the same social circles and will inevitably encounter one another and become linked themselves.
- Link prediction heuristics measure how “close” people are



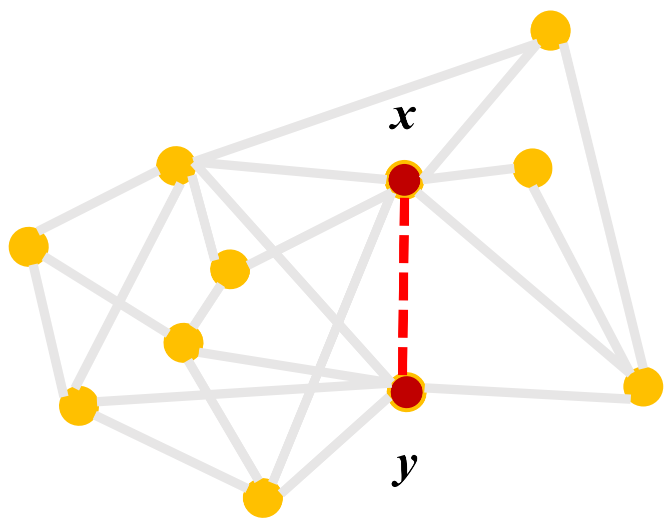
Red nodes are close to each other



Red nodes are more distant

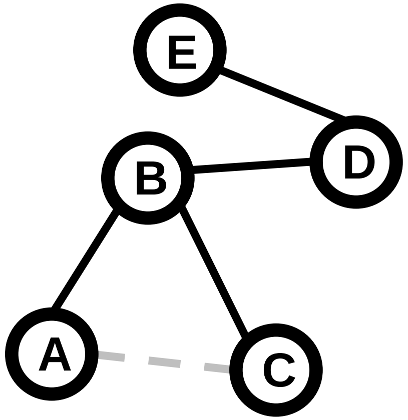
Link Prediction Methods

- Local
  - Common Neighbours (CN)
  - Jaccard's Coefficient (JC)
  - Adamic-Adar (AA)
  - Preferential attachment (PA)
  - Other:
    - Salton index
    - Sorensen index
    - Hub Promoted index
    - Hub Depressed Index
- Global
  - Path based
  - Path based Hitting time
  - SimRank



Graph Distance

- Graph distance: Length of shortest path between two nodes in the graphs
- E.g., the distance between nodes in graphs:



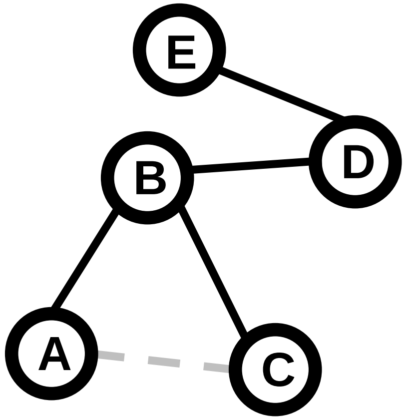
(A, C)	?
(C, D)	?
(A, E)	?

Graph Distance


```python
# Calculate graph distance
def calculate_graph_distance(G, source, target):
    return nx.shortest_path_length(G, source=source, target=target)

# Instantiate the graph
G = nx.Graph()

edges = [("A", "B"), ("B", "C"), ("B", "D"), ("D", "E")]
# add node/edge pairs
G.add_edges_from(edges)

# Use spring layout for better visualization
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True)

# Calculate distances
distances = {
    ("A", "C"): calculate_graph_distance(G, 'A', 'C'),
    ("C", "D"): calculate_graph_distance(G, 'C', 'D'),
    ("A", "E"): calculate_graph_distance(G, 'A', 'E'),
}

print("Graph distances:")
for nodes, distance in distances.items():
    print(f"{nodes} | {distance}")
```

✓ 0.0s

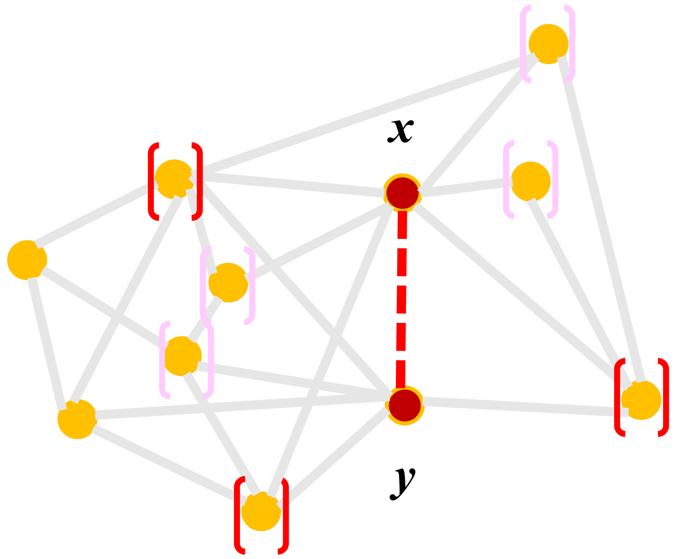
```
Graph distances:
('A', 'C') | 2
('C', 'D') | 2
('A', 'E') | 3
```



(A, C)	2
(C, D)	2
(A, E)	3

Common Neighbors

- How many neighbours are in common between x and y
- X and Y have 3 common neighbours, more likely to collaborate
- Let N(x) denote the set of nodes adjacent to x, N(x)= {m| (x , m) ∈ E}



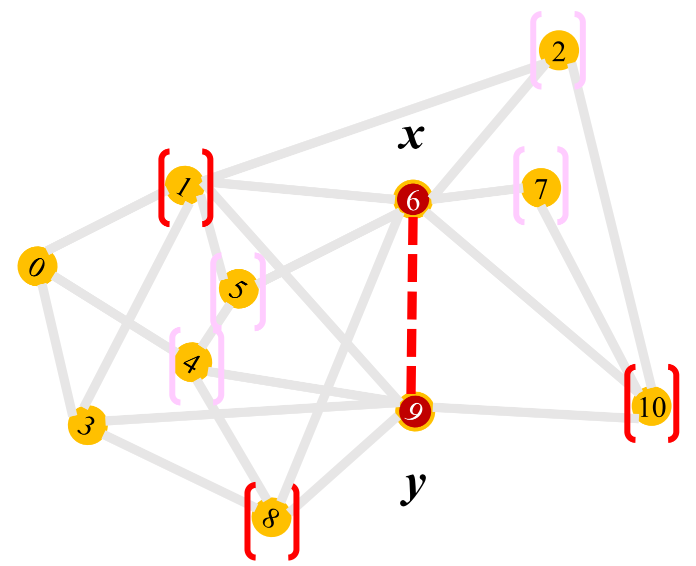
$$CN = |N(x)| \cap |N(y)| = 3$$

Common Neighbours: Sample code

```python
# Instantiate the graph
G = nx.Graph()

# Define edges
edges = [(0, 1), (1, 2), (0, 3), (1, 3), (0, 4), (1, 5), (4, 5), (1, 6),
         (2, 6), (5, 6), (6, 7), (3, 8), (4, 8), (6, 8), (3, 9), (4, 9),
         (1, 9), (8, 9), (2, 10), (6, 10), (7, 10), (9, 10)]

# Add edges to the graph
G.add_edges_from(edges)

# Use spring layout for better visualization
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True)

# Calculate common neighbors of nodes 6 and 9
cn_list = sorted(nx.common_neighbors(G, 6, 9))

# Print common neighbors
print(f"The common neighbors of nodes 6 and 9 are: {cn_list}")
```

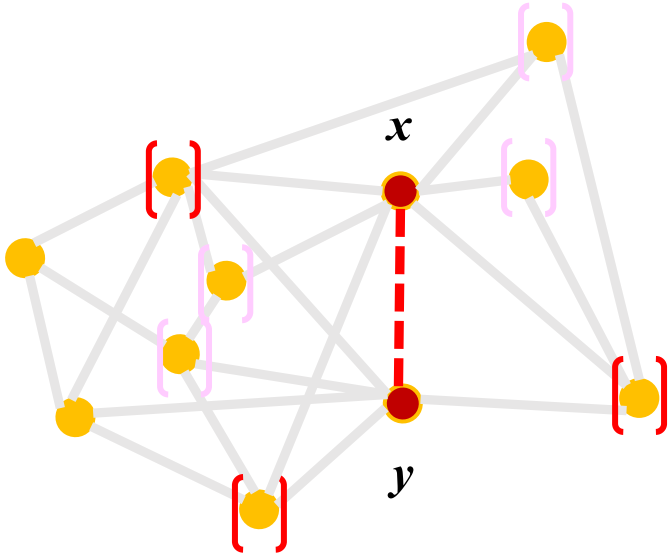
✓ 0.0s

```
The common neighbors of nodes 6 and 9 are: [1, 8, 10]
```



Jaccard's coefficient

- How likely a neighbour of x is also a neighbour of y
- Same as common neighbors, adjusted for degree



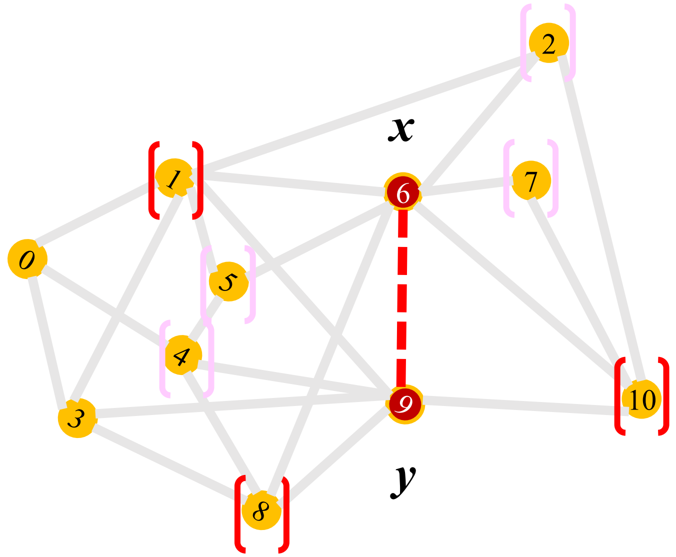
$$JC = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|} = \frac{CN}{d_x + d_y - CN}$$

Jaccard's coefficient: sample code


```python
# Define list of node pairs
node_list = [(6, 9), (1, 3), (3, 9)]

# Calculate Jaccard coefficients for each pair
coefficients = {}
preds = nx.jaccard_coefficient(G, node_list)
for u, v, p in preds:
    coefficients[(u, v)] = p

# Print Jaccard coefficients
for nodes, coeff in coefficients.items():
    print(f"The Jaccard coefficient of nodes {nodes} is: {coeff:.8f}")
```

✓ 0.0s

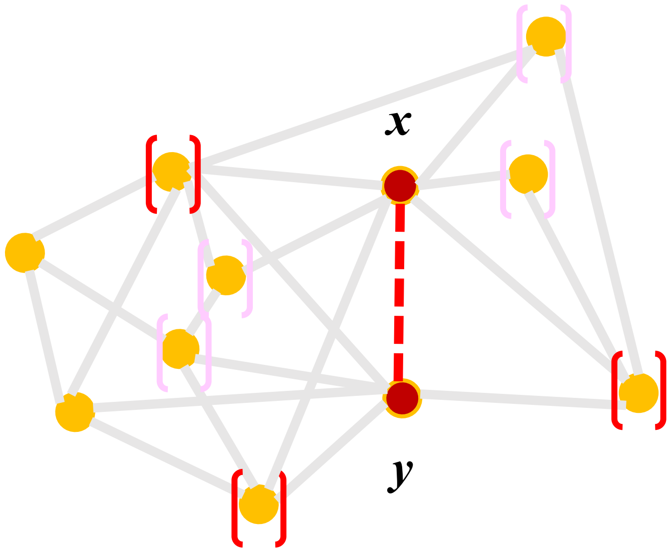
```
The Jaccard coefficient of nodes (6, 9) is: 0.37500000
The Jaccard coefficient of nodes (1, 3) is: 0.25000000
The Jaccard coefficient of nodes (3, 9) is: 0.28571429
```

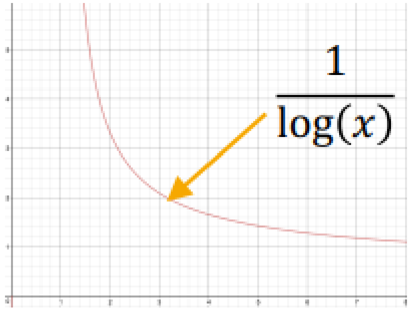


Adamic-Adar (AA)

- Large weight to common neighbours with low degree (the lower the degree the higher the relevance)
- E.g., Neighbours who are linked with 2 nodes are assigned weight = 1/log(2) = 1.4
  - Neighbours who are linked with 5 nodes are assigned weight = 1/log(5) = 0.62

$$AA = \sum_{z \in CN} \frac{1}{\log d_z}$$



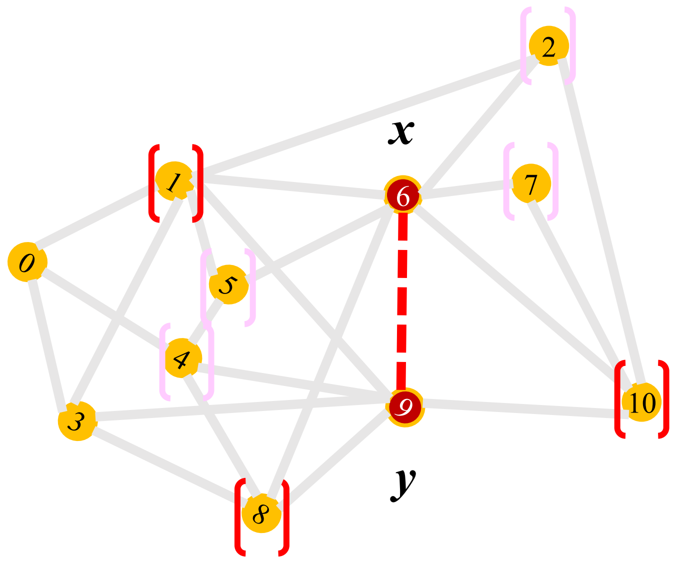


Adamic-Adar (AA): sample code

```python
# Define list of node pairs
node_list = [(6, 9), (1, 3), (3, 9)]

# Calculate Adamic-Adar indices for each pair
indices = {}
preds = nx.adamic_adar_index(G, node_list)
for u, v, p in preds:
    indices[(u, v)] = p

# Print Adamic-Adar indices
for nodes, index in indices.items():
    print(f"The Adamic-Adar index of nodes {nodes} is: {index:.8f}")
```

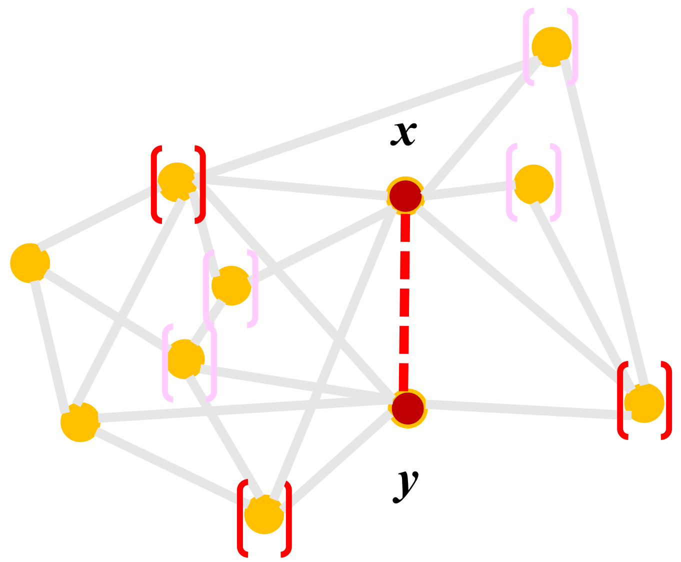
✓ 0.0s

```
The Adamic-Adar index of nodes (6, 9) is: 2.00080567
The Adamic-Adar index of nodes (1, 3) is: 1.53157416
The Adamic-Adar index of nodes (3, 9) is: 1.27945815
```



Preferential attachment (PA)

- Better connected nodes are more likely to form more links.
- The more popular a node is the more probable it will form a link with popular nodes.
- This depends on the degrees of the nodes not on their neighbourhoods



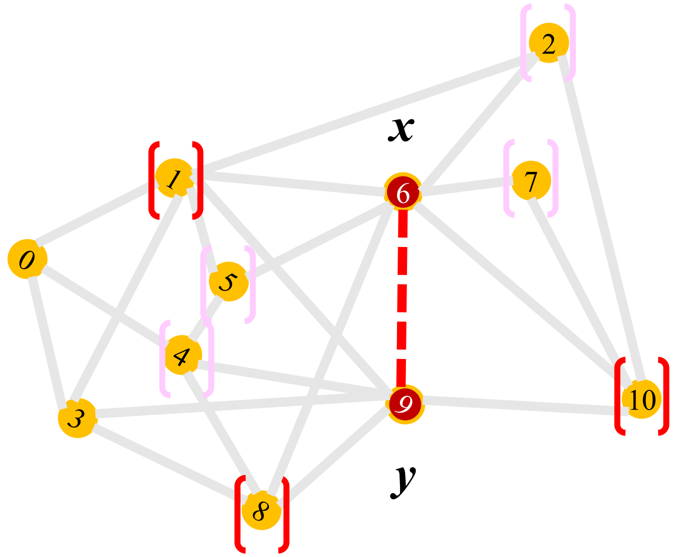
$$PA = |N(x)|.|N(y)| = d_x.d_y$$

Preferential attachment (PA): sample code


```python
# Define list of node pairs
node_list = [(6, 9), (1, 3), (3, 9)]

# Calculate Preferential Attachment scores for each pair
scores = {}
preds = nx.preferential_attachment(G, node_list)
for u, v, p in preds:
    scores[(u, v)] = p

# Print Preferential Attachment scores
for nodes, score in scores.items():
    print(f"The Preferential Attachment score of nodes {nodes} is: {score:.8f}")
```

✓ 0.0s

```
The Preferential Attachment score of nodes (6, 9) is: 30.00000000
The Preferential Attachment score of nodes (1, 3) is: 24.00000000
The Preferential Attachment score of nodes (3, 9) is: 20.00000000
```



Other neighbourhood-based methods

- Salton index

$$score(x, y) = \frac{|N(x) \cap N(y)|}{\sqrt{|N(x)||N(y)|}}$$

- Sorensen index

$$score(x, y) = \frac{2|N(x) \cap N(y)|}{|N(x)| + |N(y)|}$$

- Hub Promoted index

$$score(x, y) = \frac{|N(x) \cap N(y)|}{\min\{|N(x)|, |N(y)|\}}$$

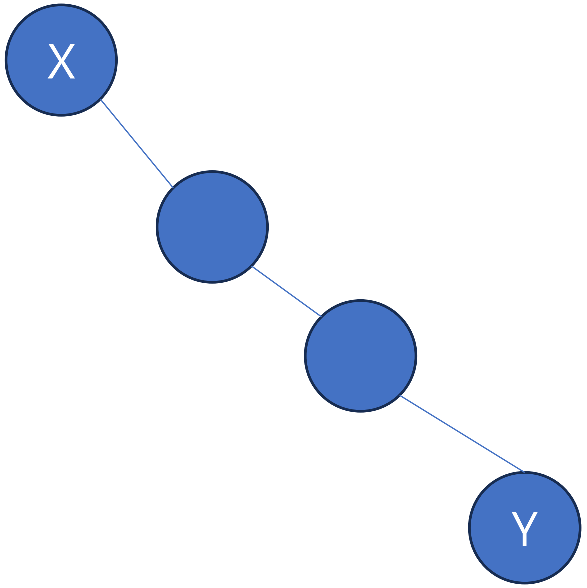
- Hub Depressed Index

$$score(x, y) = \frac{|N(x) \cap N(y)|}{\max\{|N(x)|, |N(y)|\}}$$

https://www.nature.com/articles/s41598-019-57304-y
https://www.nature.com/articles/s41598-020-76799-4

Methods for Link Prediction: Path based

- Use the (shortest) distance between two nodes as a link prediction measure
- Score(x,y) = length of shortest path between x and y.
- Very basic approach, it does not consider connections among (x,y) but only the distance



Score(X,Y) = 3

Methods for Link Prediction: Path based

- Katz index:

Element (x,y) in the adjacency matrix

$$score(x,y) = \sum_{l=1}^{\infty} \beta^l \left| paths_{xy}^{(l)} \right| = \beta A_{xy} + \beta^2 A_{xy}^2 + \dots$$

- Sum over ALL paths of length ℓ
- $0 < \beta < 1$ is a parameter of the predictor, exponentially damped to count short paths more heavily
- Small damped parameter = predictions much like common neighbours

Link Prediction Methods: Path based Hitting time

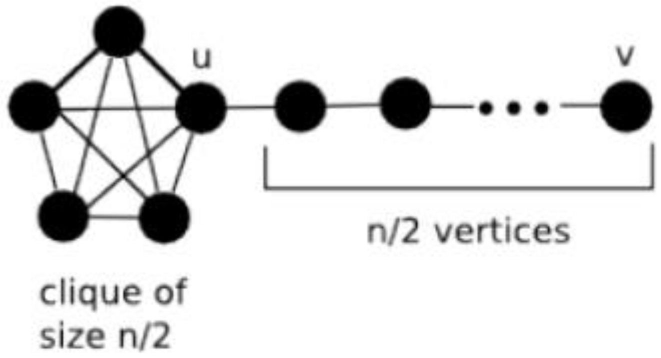
- Consider a random walk on graph G that starts at x and iteratively moves to a neighbour of x chosen uniformly random from N(x)
- **Hitting time (Hxy)** from x to y is the expected number of steps it takes for the random walk starting at x to reach y.

$$score(x,y) = -H_{x,y} = -\frac{1}{|N(\mathrm{x})|}\sum_{k}(1+H_{\mathrm{k},y})$$

$$H(i,j) = 1 + \sum_{k\sim i} p_{ik} H(k,j),\ j \neq i, \quad H(i,i) = 0.$$

- Is Hitting Time Symmetric?
  - NOT symmetric
  - E.g., path from u to v is different

  From v to u



- Intuition: Two objects are similar, if they are related to similar objects
- Two objects x and y are similar, if they are related to objects a and b respectively and a and b are themselves similar

$$similarity\ (x, y) = \frac{\sum_{a \in N(x)} \sum_{b \in N(y)} similarity(a, b)}{|N(x)|.\,|N(y)|}$$

- Expresses the average similarity between neighbours of x and neighbours of y:

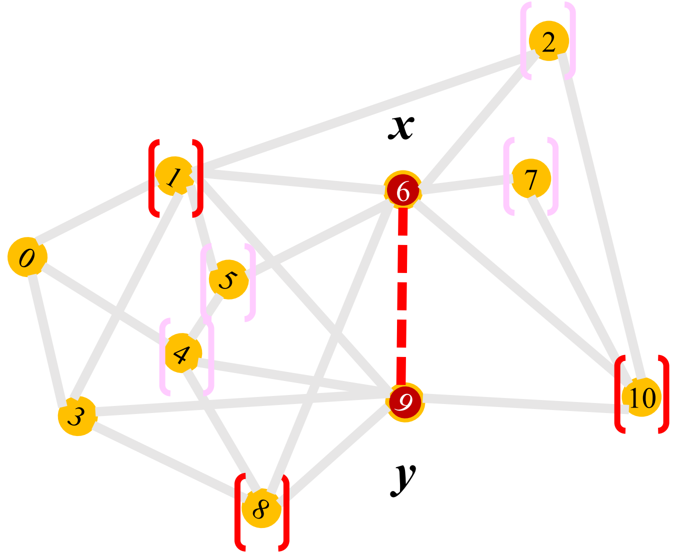
score(x, y) = similarity(x, y)

SimRank: sample code

```python
# Define source and target nodes
source = 6
target = 9

# Calculate SimRank similarity between source and target
similarity = nx.simrank_similarity(G, source, target)

# Print SimRank similarity
print(f"The SimRank similarity between nodes {source} and {target} is: {similarity}")
```

✓ 0.0s

```
The SimRank similarity between nodes 6 and 9 is: 0.4283123305678891
```



Evaluation Metrics

- Precision and Recall, F score
  - Precision = $\frac{TP}{TP+FP}$
  - Recall = $\frac{TP}{TP+FN}$
  - F = $\frac{2*Precision*Recall}{Precision+Recall}$

True Positive (TP): when both the actual and predicted values are 1.
True Negative (TN): when both the actual and predicted values are 0.
False Positive (FP): when the actual value is 0 but the predicted value is 1.
False Negative (FN): when the actual value is 1 but the predicted value is 0.

- True positive rate (TPR), False positive rate (FPR)
  - TPR = $\frac{TP}{TP+FN}$
  - FPR = $\frac{FP}{FP+TN}$
- Other metrics:
  - MAP, Precision at K,…

Software Tools

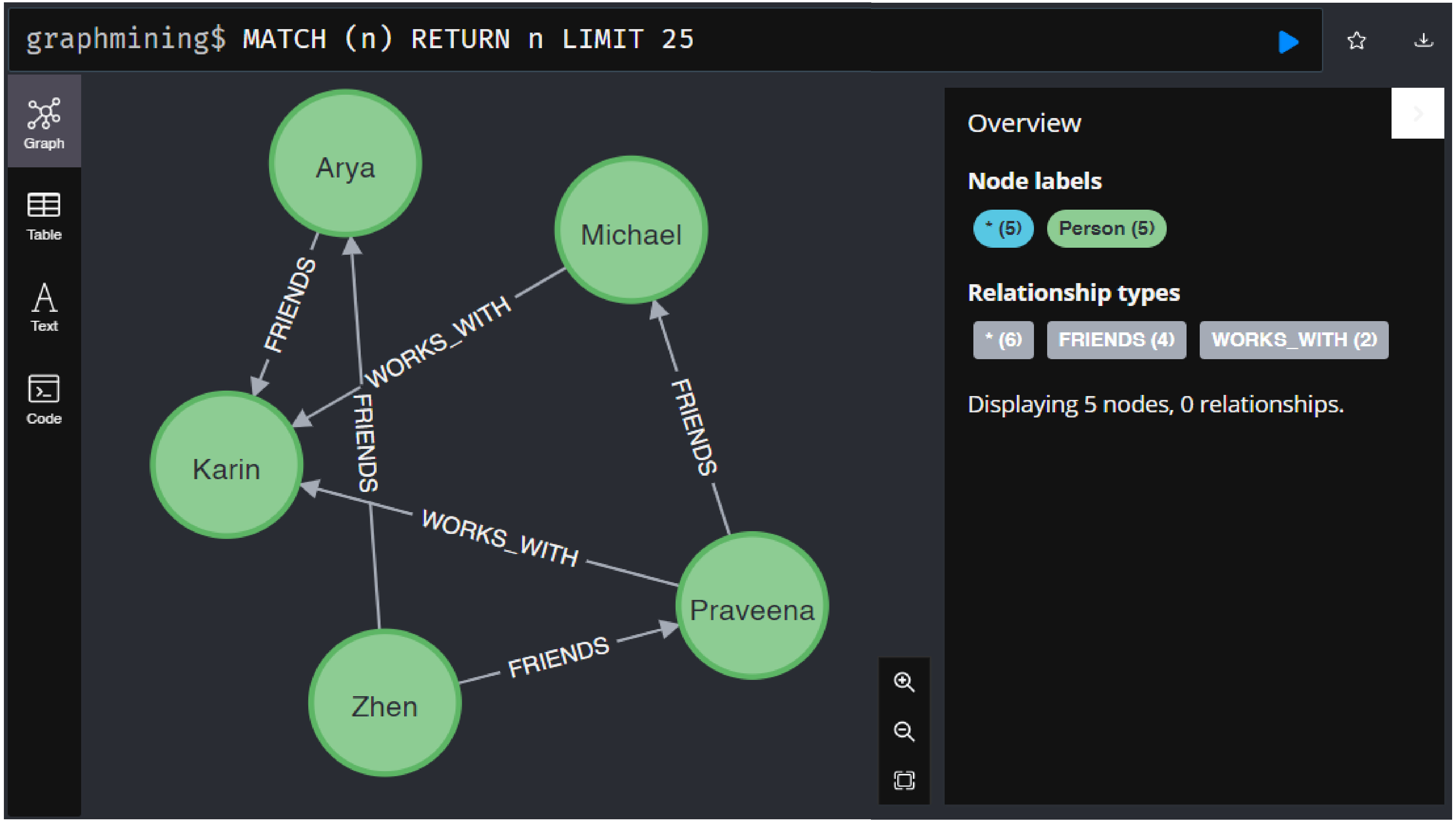
- NodeXL
  - Integrates network analysis into Microsoft Office and Excel
  - No programming skills required
  - Utilizes SNAP library
- InfoVis Cyberinfrastructure
  - Software framework for information visualization
  - Supports Linux, MacOSX, and Windows
  - Enables creation of interactive visualizations
- Analytic Technologies
  - Specialized software for social network analysis
  - Windows-based platform
  - Facilitates data preprocessing and advanced analysis

Software Tools

- Neo4j
  - Graph visualization software
  - Scalable and performance-oriented
  - Ideal for modeling and querying large-scale graph data
- NetworkX
  - Python package for complex network analysis
  - Comprehensive suite of algorithms
  - Widely used by researchers and data scientists

Software Tools: sample code

- Show database



Software Tools: sample code


```
CREATE
 (zhen:Person {name: 'Zhen'}),
 (praveena:Person {name: 'Praveena'}),
 (michael:Person {name: 'Michael'}),
 (arya:Person {name: 'Arya'}),
 (karin:Person {name: 'Karin'}),

 (zhen)-[:FRIENDS]->(arya),
 (zhen)-[:FRIENDS]->(praveena),
 (praveena)-[:WORKS_WITH]->(karin),
 (praveena)-[:FRIENDS]->(michael),
 (michael)-[:WORKS_WITH]->(karin),
 (arya)-[:FRIENDS]->(karin)
```

Table

Code

Added 5 labels, created 5 nodes, set 5 properties, created 6 relationships, completed after 21 ms.

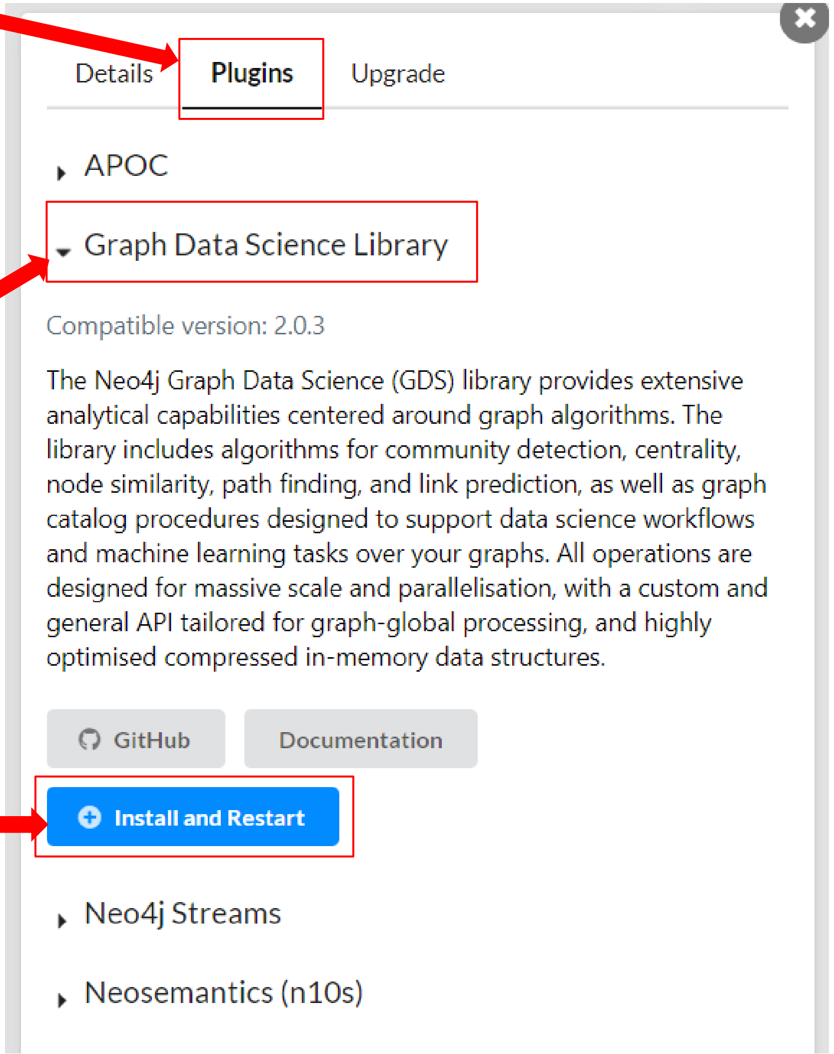
Software Tools: sample code

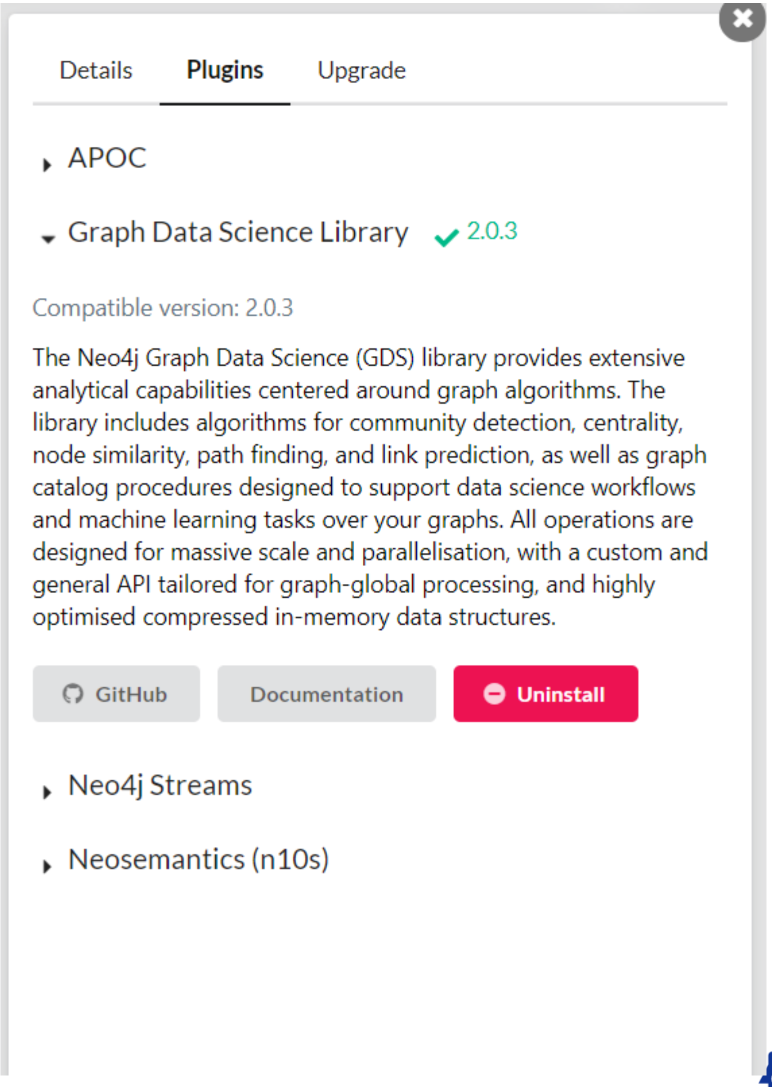
- Install the plugin to use graph algorithms

1. Click "Plugins" tab in the pane on the right.

2. Click on "Graph Data Science Library".

3. Install plugin.





Software Tools: sample code

- Calculate the number of common neighbors without considering relation type

```
MATCH (p1:Person {name: 'Michael'})
MATCH (p2:Person {name: 'Karin'})
RETURN gds.alpha.linkprediction.commonNeighbors(p1, p2) AS score
```

	score
1	1.0

- Calculate the number of common neighbors with considering relation type

```
MATCH (p1:Person {name: 'Michael'})
MATCH (p2:Person {name: 'Karin'})
RETURN gds.alpha.linkprediction.commonNeighbors(p1, p2, {relationshipQuery: "FRIENDS"})
AS score
```

	score
1	0.0

Software Tools: sample code

- Calculate the Adamic Adar without considering relation type

```
MATCH (p1:Person {name: 'Michael'})
MATCH (p2:Person {name: 'Karin'})
RETURN gds.alpha.linkprediction.adamicAdar(p1, p2) AS score
```

Table

Text

Code

"score"
0.9102392266268373

- Calculate the Adamic Adar without considering relation type

```
MATCH (p1:Person {name: 'Michael'})
MATCH (p2:Person {name: 'Karin'})
RETURN gds.alpha.linkprediction.adamicAdar(p1, p2, {relationshipQuery: 'FRIENDS'})
AS score
```

Table

Text

Code

"score"
0.0

Software Tools: sample code

- Calculate the Preferential Attachment without considering the relation type

```
MATCH (p1:Person {name: 'Michael'})
MATCH (p2:Person {name: 'Karin'})
RETURN gds.alpha.linkprediction.preferentialAttachment(p1, p2) AS score
```

Table

Text

Code

"score"
6.0

Software Tools: sample code

- Calculate the Preferential Attachment with considering the relation type

```
MATCH (p1:Person {name: 'Michael'})
MATCH (p2:Person {name: 'Karin'})
RETURN gds.alpha.linkprediction.preferentialAttachment(p1, p2,
{relationshipQuery: "FRIENDS"}) AS score
```

```
╒═══════╕
│"score"│
╞═══════╡
│1.0    │
└───────┘
```

네트워크 과학 연구실
NETWORK SCIENCE LAB

가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA