Classical Link Prediction in Complex Networks

Giacomo Fiumara

Introduction

Motivation

- Link prediction is a key problem in complex networks.
- It asks: **Can we infer which links are likely to appear, or are currently missing, given the structure of a network?**

Motivation

Why is this important?

- Understanding network growth dynamics (e.g., social network evolution).
- Discovering unseen connections in biological systems (e.g., predicting protein-protein interactions).
- Recommending new connections/products in commercial platforms (e.g., suggesting new friends or items).

Real-World Applications

Link prediction is used in:

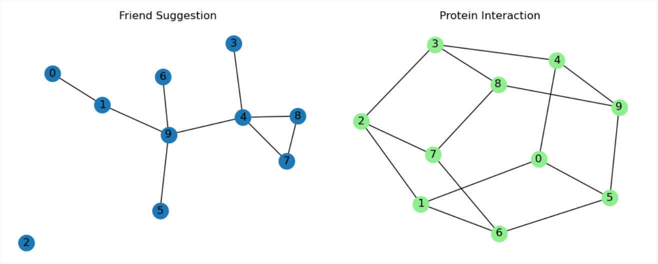
- **Social Networks:** Suggesting connections on Facebook/LinkedIn by estimating which users may know each other.
- **Biological Networks:** Identifying potential interactions between genes or proteins that have not been experimentally detected.

Real-World Applications

Link prediction is used in:

- **Recommendation Systems:** Recommending items, movies, or friends based on user-item interactions.
- **Citation Networks:** Predicting future scientific collaborations or paper citations.
- These domains share common principles: using the observable network structure to forecast new or missing links.

Real-World Applications



Problem Definition

What is Link Prediction?

Definition: Link prediction aims to infer the existence of edges that are missing from a known graph, or forecast edges that may appear in the future as the graph evolves.

Formal Problem Statement:

- Given a graph $G = (V, E)$ where V is the set of nodes and E the set of observed edges,
- Find node pairs (u, v) such that $(u, v) \notin E$, but the addition of (u, v) is likely given the structure of G.
- **Goal:** Rank all non-observed links by their likelihood, and select the top candidates for prediction or recommendation.

Types of Link Prediction Tasks

Link prediction tasks vary by the network's domain and timescale:

1. **Static Link Prediction:** Identify missing edges in a fixed snapshot of the network.
2. **Temporal/Future Link Prediction:** Predict which new edges will appear over time in a dynamic network.
3. **Heterogeneous and Attributed Networks:** Predict links in bipartite (user-item), multilayer, or attribute-rich graphs.
4. For each scenario, specialized methods adapt to the underlying data and prediction goals.

Evaluation Metrics

Evaluating Link Prediction: Why and How?

Why do we need metrics?

- Link prediction produces, for every possible non-observed edge (u, v), a score indicating how likely it is.
- To quantify the performance of our prediction algorithm, we need rigorous metrics. The choice of metric depends on:

Evaluating Link Prediction: Why and How?

Why do we need metrics?

- The application area (e.g., recommendations, scientific discovery)
- Whether we treat prediction as ranking or binary classification
- The class balance (many more negative than positive examples!)

Confusion Matrix and Basic Metrics

Confusion matrix:

	Predicted: Link	**Predicted: No Link**
True: Link	True Positive (TP)	False Negative (FN)
True: No Link	False Positive (FP)	True Negative (TN)

Assess performance using:

- True Positives (TP): Correctly predicted links
- False Positives (FP): Predicted links that do not exist
- False Negatives (FN): Missed true links
- True Negatives (TN): Correctly predicted absent links

Precision, Recall, F1-Score

Precision (Fraction of predicted links that are true):

$$Precision = \frac{TP}{TP + FP}$$

Recall (Fraction of true links actually predicted):

$$Recall = \frac{TP}{TP + FN}$$

Precision, Recall, F1-Score

F1-Score (Harmonic mean of precision and recall—balances both aspects):

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Confusion Matrix Illustration

	Pred: Link	**Pred: No Link**
True: Link	**TP** True Positive	**FN** False Negative
True: No Link	**FP** False Positive	**TN** True Negative

Accuracy: Caution Required!

Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Fraction of total predictions that are correct.

However:

- For link prediction, *most possible pairs are not true links*, making TN (True Negatives) very large.
- That means accuracy may not reflect model quality well! Prefer Precision and Recall.

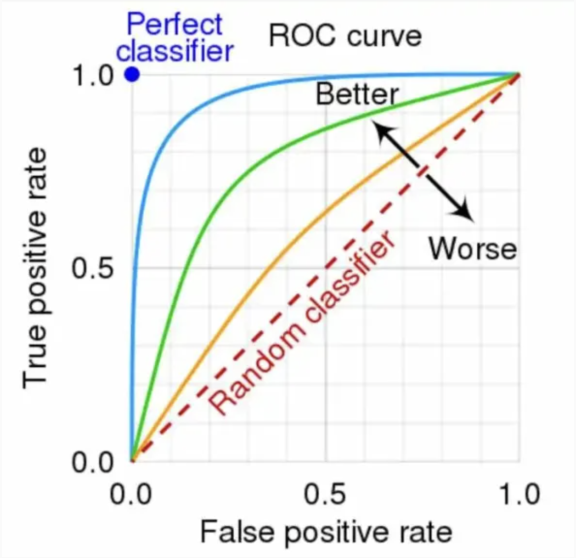
ROC Curve and Area Under Curve (AUC)

ROC Curve: Plots True Positive Rate (TPR) vs. False Positive Rate (FPR) as you vary your threshold.

$$TPR = Recall = \frac{TP}{TP + FN}$$
$$FPR = \frac{FP}{FP + TN}$$

AUC: Area under the ROC curve is a single value summarizing discrimination power.
Higher AUC (≈ 1) means better separation between positive and negative links.

ROC Curve and Area Under Curve (AUC)



Ranking Metrics: Precision@k

- Many real-world scenarios care about **top-k** predictions (e.g., friend suggestions).
- **Precision@k:**

$$Precision@k = \frac{\text{Number of true links in top k predictions}}{k}$$

- For example, if 3 out of the top 10 predicted links are correct, Precision@10 $= 0.3$.

What Are Ranked Predictions?

Most link prediction algorithms provide, for each query (e.g., a node), a **ranked list of predicted links**.

- Each candidate link is scored by the algorithm, indicating its likelihood to exist (or be added).
- The candidate links are sorted from most likely to least likely.
- Example: For node Q, we might rank all possible non-adjacent pairs (Q, v) according to their scores.
- **Goal:** Relevant (ground-truth) links should appear high—at the top—of the prediction list.

Queries and Ground Truth

In link prediction, a **query** is typically a node (or node pair) for which we want to predict missing links.

- Each query has an associated set of ground-truth relevant links (edges that should exist).
- The performance for each query is measured by how well the ranked predictions match the ground truth.

Example:

- Query: node Q
- Ground-truth missing links: (Q, A), (Q, C)

Definition of Average Precision (AP) and MAP

Average Precision (AP):

- For one query, AP averages the precision at ranks where relevant links appear in the prediction list.
- Let $N_{rel}(q)$ be the number of relevant links for query q, n_q be the number of returned predictions, $Precision@k(q)$ be the precision at rank k, and $rel_k(q)$ be 1 if rank k is relevant, 0 otherwise:

$$AP(q) = \frac{1}{N_{rel}(q)} \sum_{k=1}^{n_q} Precision@k(q) \cdot rel_k(q)$$

Definition of Average Precision (AP) and MAP

Mean Average Precision (MAP):

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(q)$$

where Q is the number of queries.

MAP Example

Suppose we have one query:

Query: node Q

Ground-truth relevant links:

$$\{(Q, A),\ (Q, C)\}$$

These are the true missing links our algorithm should ideally predict for Q.

Predicted ranked list for Q:

$$[(Q, B),\ (Q, A),\ (Q, C),\ (Q, E)]$$

MAP Example

Calculate AP for Q:

1. Rank 1: (Q, B) — not relevant, $\mathit{Precision}@1 = 0/1 = 0$
2. Rank 2: (Q, A) — relevant, $\mathit{Precision}@2 = 1/2 = 0.5$
3. Rank 3: (Q, C) — relevant, $\mathit{Precision}@3 = 2/3 \approx 0.667$
4. Rank 4: (Q, E) — not relevant, not counted for AP

Only ranks where the predicted link matches a ground-truth relevant link ((Q, A) or (Q, C)) contribute.

$$AP(Q) = \frac{0.5 + 0.667}{2} \approx 0.583$$

MAP: Interpretation

- MAP provides a single summary value for how well the algorithm ranks relevant missing links high for all queries.
- A higher MAP value means that correct links appear near the top positions across queries.
- MAP is widely used in link prediction, information retrieval, and recommender systems—it rewards both early correct predictions and overall accuracy!

Precision and Recall Example

Example scenario:

- True Links to Predict: $\{$(A,B), (B,C)$\}$
- Predicted Top Links: $\{$(A,B), (C,D)$\}$

Calculating Precision & Recall:

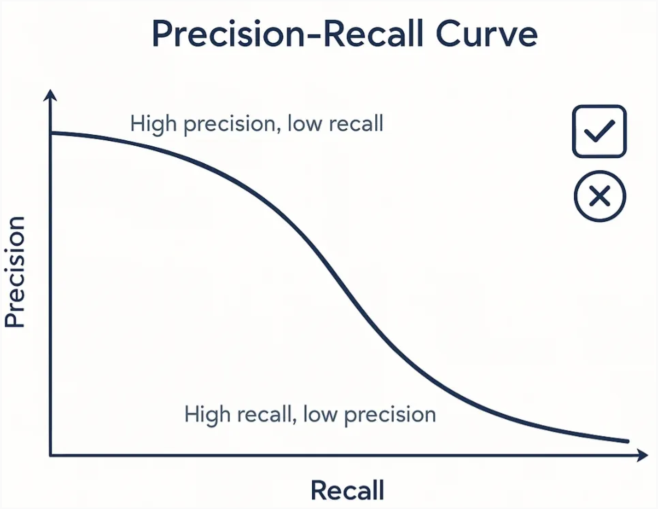
- Precision $=$ 1 correct out of 2 predicted $= \frac{1}{2}$
- Recall $=$ 1 correct out of 2 true $= \frac{1}{2}$

Precision-Recall Curve

For very imbalanced data, (as in link prediction), Precision-Recall curves are often more informative.

- Precision is plotted as a function of recall as the decision threshold varies.
- Shows trade-off: more recall generally means lower precision.

Precision-Recall Curve



Interpreting the Precision-Recall Curve

What is the Precision-Recall curve?

- It plots **Precision** (y-axis)—the fraction of predicted links that are relevant—versus **Recall** (x-axis)—the fraction of relevant links that are recovered—across different prediction score thresholds.

- Each point represents a trade-off between retrieving more links (higher recall) and being accurate in what you retrieve (higher precision).

Interpreting the Precision-Recall Curve

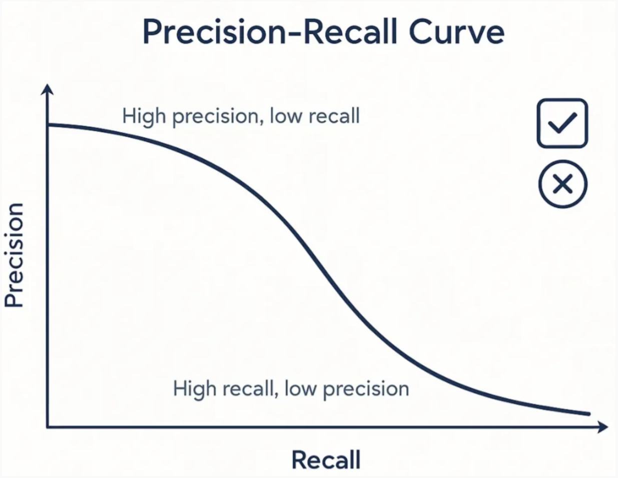
- A **high curve** (closer to the top-right) means the algorithm can recover many relevant links without including many false links.
- **Precision drops** as recall increases: the more candidates are retrieved, the more the false positives, so precision may decrease.
- **Area Under the Curve (AUC-PR)**: A larger area means better performance—many relevant links are predicted with high accuracy.
- Especially useful for **imbalanced data** (few positives, many negatives)—typical in link prediction.

Interpreting the Precision-Recall Curve

Practical interpretation:

- If curve is above the baseline (random guessing), your method is identifying relevant links better than chance.
- Steep initial rise, then gradual decline, usually reflects strong early ranking (top predictions are correct).
- Use the curve to decide thresholds for operational use—e.g., only recommend links above a precision of 0.8.

Precision and Recall Example



Taxonomy

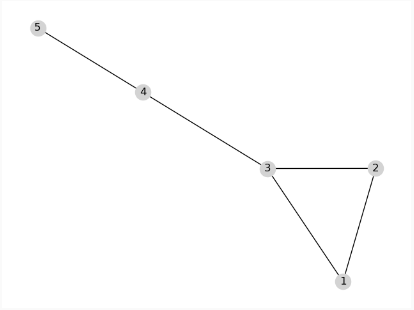
Overview of Link Prediction Methods

Classical approaches can be categorized by the level of information used:

- **Local Methods:** Only use direct neighbors of candidate nodes.
- **Global Methods:** Use the entire graph topology—e.g., paths/flows throughout the network.
- **Quasi-Local:** Combine local and global features, bridging the gap.
- **Machine Learning:** Learn complex patterns from features; discussed in separate lectures.

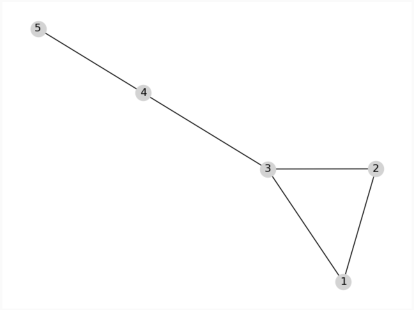
Example Network for Prediction

- Consider the network below, where several candidate links (e.g., dotted edges) are not present.
- Our goal: estimate how likely each missing link is to exist based on the visible structure.



Example Network for Prediction

- Each candidate (e.g., $(3, 5)$) will be scored differently by different methods.



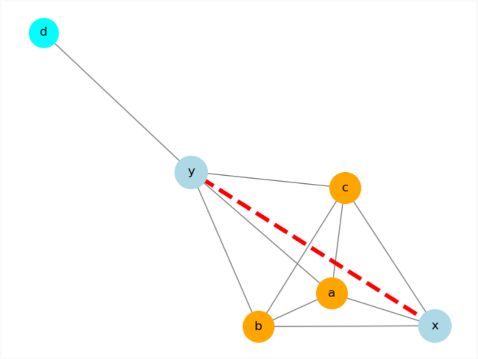
Classical Heuristics for Link Prediction

Local Heuristics: Common Neighbors

One of the simplest and most widely used link prediction methods is the **Common Neighbors** score.

- For two nodes x and y, count the number of nodes to which both are connected.
- Formula:
  $$s_{CN}(x, y) = |\Gamma(x) \cap \Gamma(y)|$$
  where $\Gamma(x)$ is the set of neighbors of x.
- Intuition: Pairs with more common neighbors are likelier to form a link.

Local Heuristics: Common Neighbors



Local Heuristics: Jaccard Coefficient

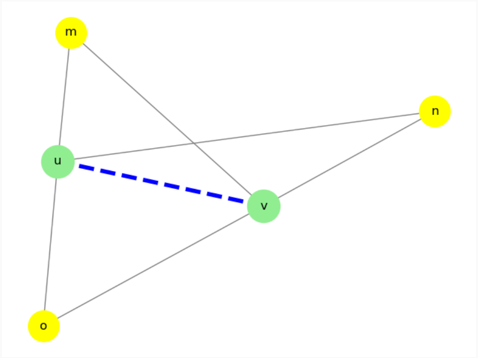
The **Jaccard Coefficient** quantifies similarity by dividing common neighbors by all possible neighbors.

- Formula:

$$s_{Jaccard}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

- Values range from 0 (no shared neighbors) to 1 (identical neighbor sets).
- Useful for normalizing in heterogeneous settings where node degrees vary widely.

Local Heuristics: Jaccard Coefficient

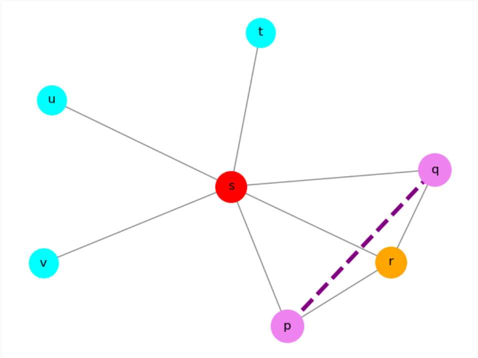


Local Heuristics: Adamic-Adar Index

The **Adamic-Adar** score weights common neighbors inversely by their degree:

- Formula:
$$s_{AA}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$$
- Rare/common neighbors contribute less/more to the score.
- This helps prioritize informative bridges rather than high-degree hubs.

Local Heuristics: Adamic-Adar Index



Local Heuristics: Preferential Attachment

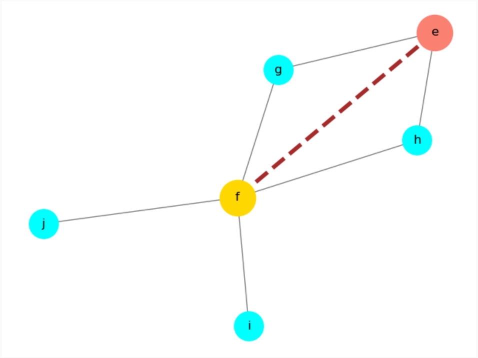
The **Preferential Attachment** index models the “rich get richer” phenomenon.

- Formula:

$$s_{PA}(x, y) = |\Gamma(x)| \times |\Gamma(y)|$$

- Assumes that new links are most likely to attach to high-degree nodes.
- Particularly relevant in growing, scale-free networks.

Local Heuristics: Preferential Attachment



Local Heuristics: Resource Allocation Index

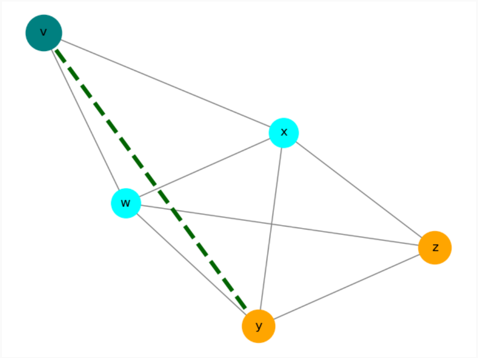
The **Resource Allocation (RA)** index is similar to Adamic-Adar but without the logarithm.

- Formula:

$$s_{RA}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{|\Gamma(z)|}$$

- Rewards connections through low-degree neighbors—favoring rare intermediaries even more strongly.
- Effective for networks with many low-degree nodes.

Local Heuristics: Resource Allocation Index

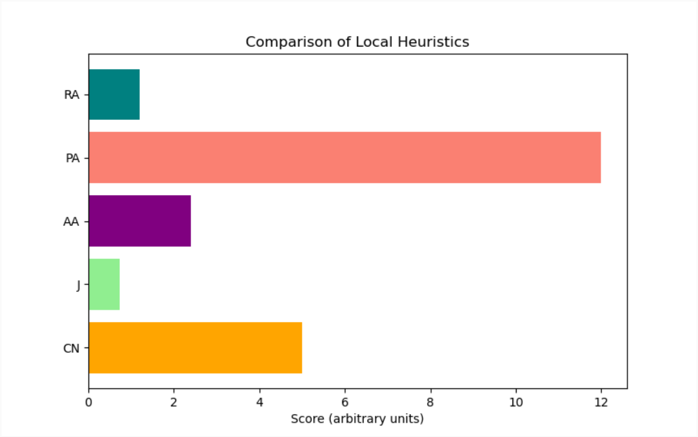


Comparing Local Heuristics

These measures differ mainly in **how they weigh common neighbors**, normalization, and their sensitivity to node degree:

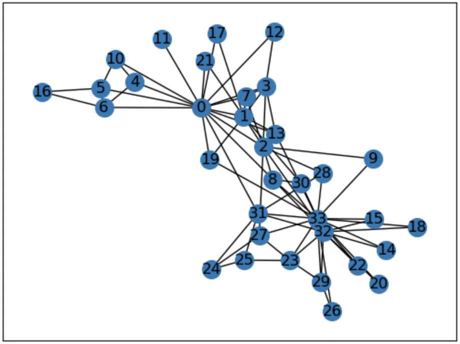
- **Common Neighbors:** Simple count; ignores degree.
- **Jaccard:** Normalizes by total neighbor set.
- **Adamic-Adar/RA:** Downweight high-degree common neighbors.
- **Preferential Attachment:** Focuses solely on source/target degrees, not on shared neighbors.

Comparing Local Heuristics



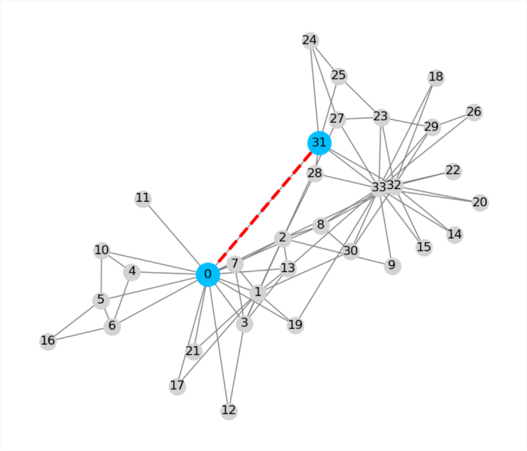
Python Example: Common Neighbors

```python
import networkx as nx
import matplotlib.pyplot as plt
G = nx.karate_club_graph()
nx.draw_networkx(G, node_size=200)
plt.show()
score = len(list(nx.common_neighbors(G, 0, 31)))
print("Common neighbors between 0 and 31: ", score)
```

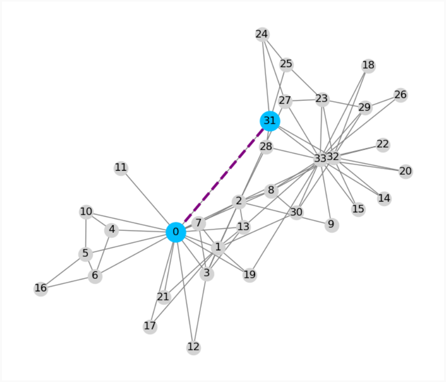


```
Common neighbors between 0 and 31:  0
```

Python Example: Common Neighbors



Python Example: Adamic-Adar Index

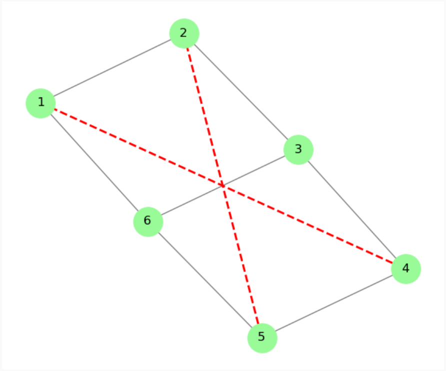


Local vs Global: Limitations

Local heuristics are fast and interpretable, but have limitations:

- Sensitive to local network density; may miss global structure.
- Prone to favoring high-degree nodes.
- Can't detect multi-hop paths and community structure.
- For more challenging cases, we need global and quasi-local methods!

Local vs Global: Limitations

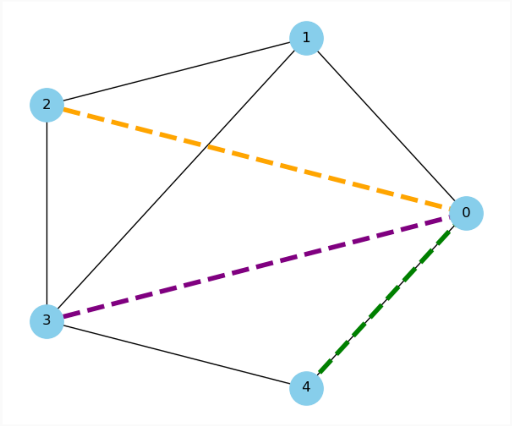


Visual Comparison: Example Rankings

A demonstration of how different local heuristics rank candidate non-adjacent pairs in a small network.

- (Q, A): Highest score with Common Neighbors / Adamic-Adar
- (Q, E): Lower score, fewer or non-informative shared neighbors
- Visualizing these rankings helps understand method biases.

Visual Comparison: Example Rankings



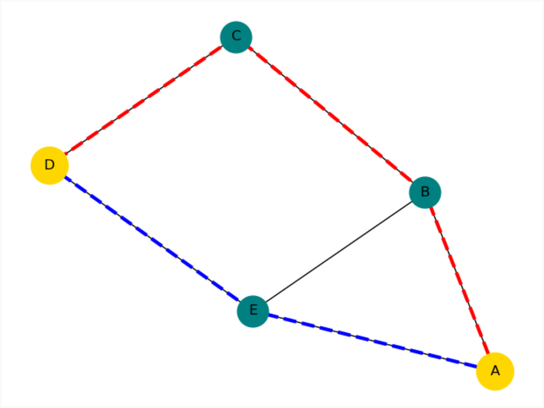
Global and Quasi-local Methods

Global Methods: Katz Index

The **Katz Index** considers all possible paths between two nodes—giving exponentially decreasing weight to longer paths.

- Formula:
$$s_{Katz}(x, y) = \sum_{\ell=1}^{\infty} \beta^{\ell} \, |\mathcal{P}_{\ell}(x, y)|$$
where $\mathcal{P}_{\ell}(x, y)$ is the set of all paths of length ℓ from x to y, and β is a damping parameter ($0 < \beta < 1$).
- Paths of all lengths matter, but short paths matter much more.
- Useful for discovering indirect connections and latent structure.

Illustration: Katz Index



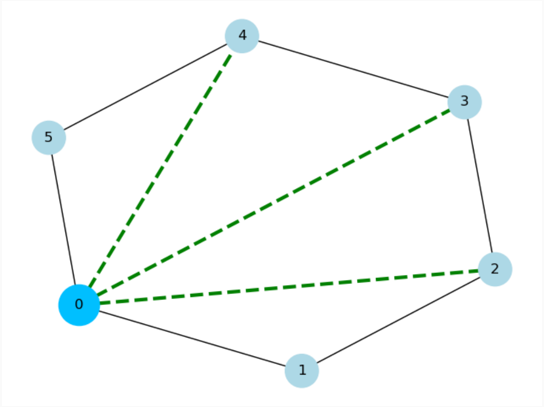
Visualization of short and long paths between candidate nodes.

Global Methods: Rooted PageRank

Rooted PageRank simulates a random walk starting from one node: the probability it lands on another node is used as their link prediction score.

- Captures “reachability” via random walks, accounting for both local and global structure.
- Often used in web link prediction and graph-based recommendation.
- Balances bias toward close nodes and diffuse connectivity.

Illustration: Rooted PageRank



Random walks starting from the candidate node, highlighting probabilistic paths.

Global Methods: SimRank

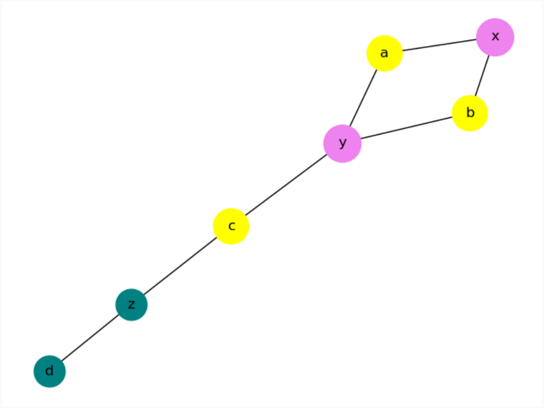
SimRank (SR) is based on the intuition: "Two nodes are similar if their neighbors are similar."

- Recursive definition:

$$s_{SR}(x, y) = \begin{cases} 1 & \text{if } x = y \\ \frac{C}{|\Gamma(x)||\Gamma(y)|} \sum_{a \in \Gamma(x)} \sum_{b \in \Gamma(y)} s_{SR}(a, b) & \text{if } x \neq y \end{cases}$$

- C: decay factor ($0 < C < 1$).
- Useful for similarity-based link prediction in large graphs.

Illustration: SimRank



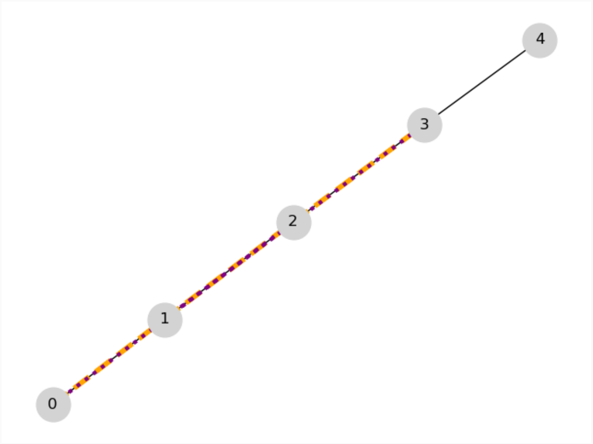
Visualizes recursive neighbor similarity computation for two nodes.

Quasi-local Hybrid Methods

Some advanced heuristics mix local and global ideas:

- Local Path Index: Weighs paths of length 2 and 3 for candidate links.
- Leicht-Holme-Newman (LHN) Index: Normalizes by expected value under random graph assumptions.
- Balances computational speed with information depth.

Illustration: Quasi-local Hybrid Methods



Hybrid: Emphasizes both short and medium-length paths.

Python Example: Katz Index

Computing Katz index in Python:

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.complete_graph(5)
nx.draw_networkx(G, node_size=200)
plt.show()
scores = nx.katz_centrality_numpy(G, alpha=0.005, beta=1.0)
print("Katz index scores:", scores)
```

Global Methods: Strengths and Limitations

Global and quasi-local heuristics offer:

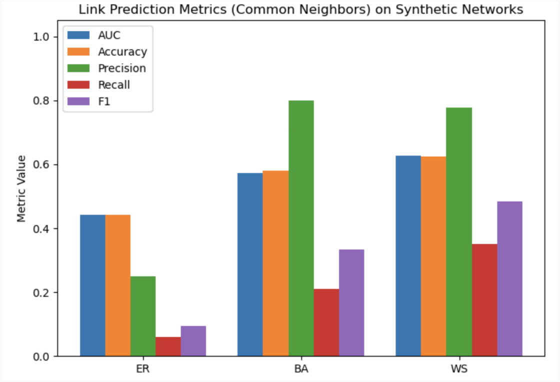
- Strength: Capture multi-hop and latent relationships.
- Strength: Can find links outside densely connected regions.
- Limitation: Higher computational cost (often $O(N^3)$ or more).
- Limitation: Need parameter tuning that affects results.

Benchmarking Methods

To compare link prediction methods, use standard benchmarks:

- Test on synthetic networks (ERN, BA, WS) and real datasets.
- Evaluate using precision, recall, AUC, MAP, PR curves.
- Cross-validation ensures robustness to overfitting.

Visual: Benchmarking and Metrics



Network Topology Determines Heuristic Effectiveness

Differences in link prediction metrics across ER, BA, WS networks are driven by their graph structure:

- Heuristics like **Common Neighbors** exploit clustering and neighborhood redundancy.
- Network models vary in clustering, degree distribution, and the presence of hubs.
- As a result, the same method yields different accuracy, precision, recall, F1, and AUC depending on network type.

Erdős–Rényi (ER) Networks

- **Structure:** Random edges, low clustering, flat degree distribution.
- **Effect:** Common neighbors occur mostly by chance—not predictive of true links.
- **Performance:** Lower accuracy, precision, recall, F1, and AUC.

Barabási–Albert (BA) Networks

- **Structure:** Scale-free, hubs with high degree, low-to-medium clustering.
- **Effect:** Links around hubs have many common neighbors—good for prediction, but can produce false positives.
- **Performance:** Intermediate scores; precision often higher around hubs.

Watts–Strogatz (WS) Networks

- **Structure:** Lattice with random shortcuts, high clustering, regular degree.
- **Effect:** Most links are between clustered neighbors—common neighbors reliably indicate missing links.
- **Performance:** Highest metrics; ideal structure for heuristics exploiting local redundancy.

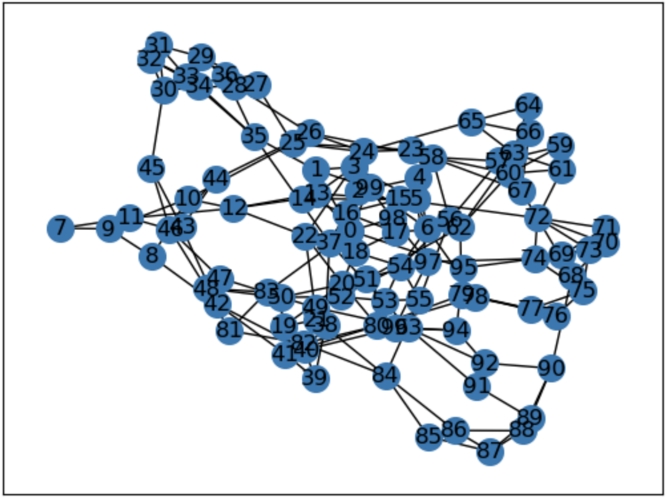
Summary Table: Structures and Performance

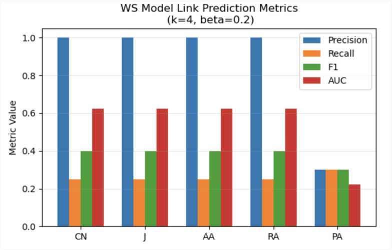
Network	**Clustering**	**Hubs**	**Signal for CN**	**Performance**
ER	Low	No	Weak	Low
BA	Low/Med	Yes	Medium	Moderate
WS	High	No	Strong	High

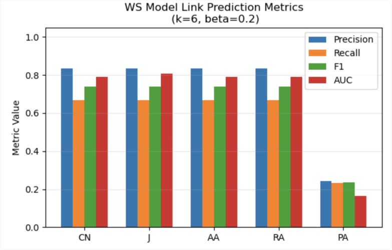
Key Insights

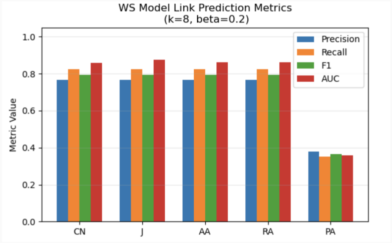
- **Common Neighbors** performs best in networks with high clustering (WS).
- Hub structure (BA) boosts scores for certain pairs, but can also mislead.
- Methods insensitive to topology (ER) generally yield weak predictions.

```
--- Watts–Strogatz: k = 4 ---
```




WS Model Link Prediction Metrics
(k=4, beta=0.2)
Metric Value
1.0
0.8
0.6
0.4
0.2
0.0
CN
J
AA
RA
PA
Precision
Recall
F1
AUC


WS Model Link Prediction Metrics
(k=6, beta=0.2)
Metric Value
1.0
0.8
0.6
0.4
0.2
0.0
CN
J
AA
RA
PA
Precision
Recall
F1
AUC


WS Model Link Prediction Metrics
(k=8, beta=0.2)
Metric Value
1.0
0.8
0.6
0.4
0.2
0.0
CN
J
AA
RA
PA
Precision
Recall
F1
AUC

Low k Means Signal Is Strongest for Local Heuristics

- For $k = 2$, the network is very **sparse**—each node has only 2 neighbors.
- In such scarce conditions, **sharing a neighbor is a rare and strong signal** of structural closeness.
- Most candidate node pairs have 0 common neighbors, making a positive prediction highly reliable.

Why Performance Drops for Higher k

- As k (average degree) increases, the network becomes denser.
- **More pairs share neighbors just by chance**, even if not truly close in the network.
- This clutter reduces the discriminative power of heuristics like Common Neighbors and Adamic-Adar.
- Precision and recall suffer due to increased false positives.

Summary: Signal-to-Noise and Predictive Power

- In **sparse, highly clustered** networks, correctly predicted links stand out.
- In **dense networks**, local similarity is less meaningful—link prediction becomes harder.
- **Practical takeaway:** Simple heuristics work best in sparse, local, structured settings—real-world dense graphs require more sophisticated methods.

Limitations of Classical Link Prediction

Despite usefulness, classical methods have clear boundaries:

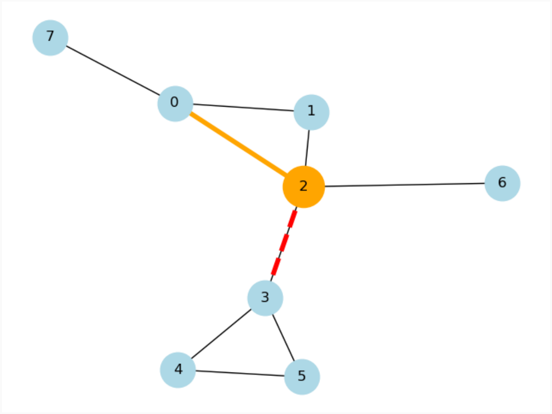
- Do not use node or edge attributes.
- Can't capture patterns beyond pure topology.
- Often fail for multiplex, heterogeneous, or dynamic temporal networks.
- Modern graph learning methods address these gaps!

Pitfalls and Modern Extensions

Classical Link Prediction Limitations: Common Pitfalls

- **Bias toward high-degree nodes:** Heuristics like Preferential Attachment always favor hubs.
- **Failure with weakly clustered networks:** Local methods perform poorly when clustering is low.
- **Insensitive to edge/node attributes:** No use of real-world information beyond topology.
- **Vulnerable to network perturbations:** Minor edge changes can shift rankings unpredictably.

Illustration: Failure Cases in Classical Prediction



Classical heuristics missing key links or misranking candidates.

Summary Table: Local and Global Methods

Method	**Type**	**Pros**	**Cons**
CN	Local	Simple, Fast	Ignores global
J	Local	Normalized	Degree sensitive
AA	Local	Downweights hubs	Still local only
PA	Local	Hubs detection	Ignores clustering
Katz, RPR, SimRank	Global	Indirect links	Computationally heavy

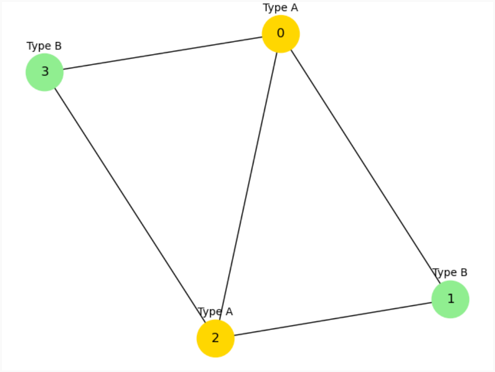
When Do We Need More?

- Real networks have node features: age, category, profession, interests, etc.
- Many real-world patterns depend on temporal, attribute, or multiplex effects.
- Classical heuristics cannot incorporate these sources of information.
- **Modern methods leverage attributes, time, and multilayer structure.**

Modern Approaches: Attribute-Assisted and Learning-Based

- Attribute-based methods: combine features (e.g., similarity measures for profile data) with topology.
- Machine learning (ML): train classifiers (logistic regression, trees, etc.) using topological and attribute-based features.
- Graph neural networks (GNNs): deep learning methods exploiting neighborhood aggregation.

Illustration: Attribute-Augmented Link Prediction

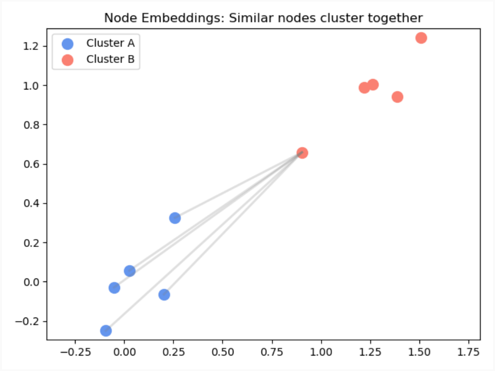


Node attributes supplementing topological link predictors.

Embeddings: The New Frontier

- Node embedding methods (e.g., node2vec, DeepWalk) map nodes to a vector space.
- Distances in this space reflect likelihood of link formation.
- Embedding features can be used in any ML model for flexible prediction.

Illustration: Link Prediction via Node Embeddings



Visual: Vector embeddings clustering likely pairs.

Classical vs Modern: A Comparative Perspective

- Classical heuristics: interpretable, fast, require only the adjacency matrix.
- Attribute-augmented and ML methods: higher accuracy, handle more complex patterns, less interpretable.
- Deep learning: best for large, complex, richly annotated networks, but hardest to interpret.
- Choice depends on data, goals, and interpretability needs.

Outlook: Where Next?

- Ongoing research: explainable ML, temporal/multilayer graphs, transfer learning, fairness.
- Best practices: compare classical, shallow ML, and deep learning on your networks.
- Keep combining theory, simulation, and real data!