Mining Graph Data

LINK PREDICTION

Lecturer: Le Ngoc Thanh

Email: lnthanh@fit.hcmus.edu.vn

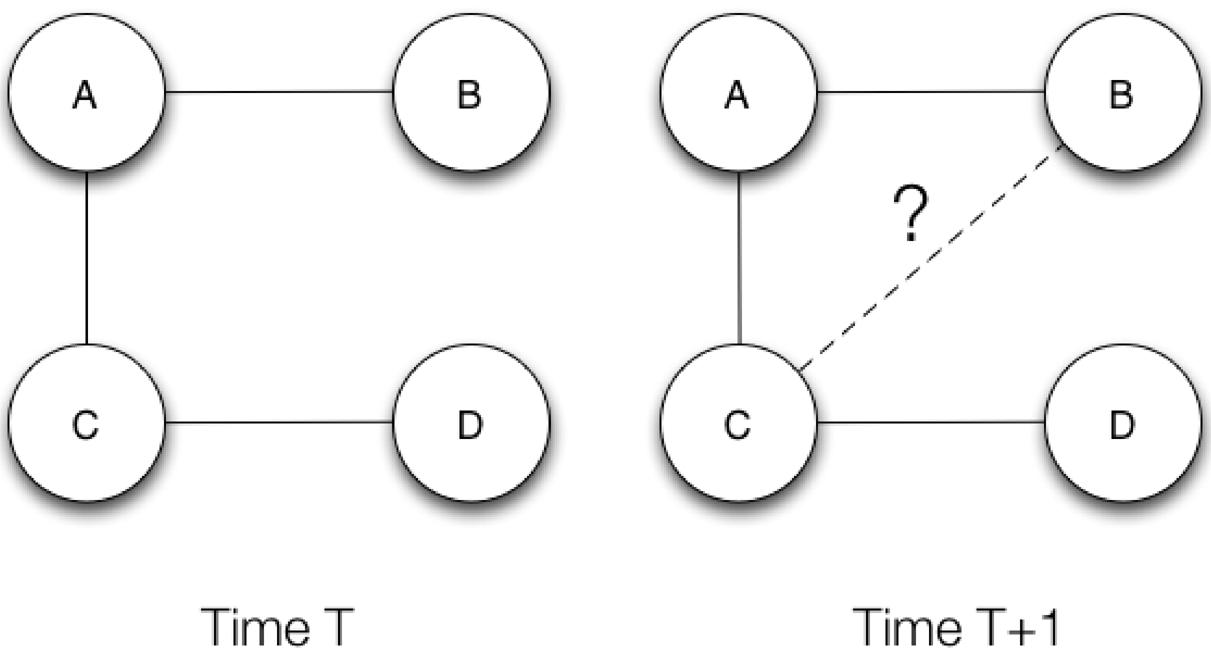
fit@hcmus

Content

- **Link prediction**
- Learn in link prediction
  - Unsupervised learning
  - Supervised learning

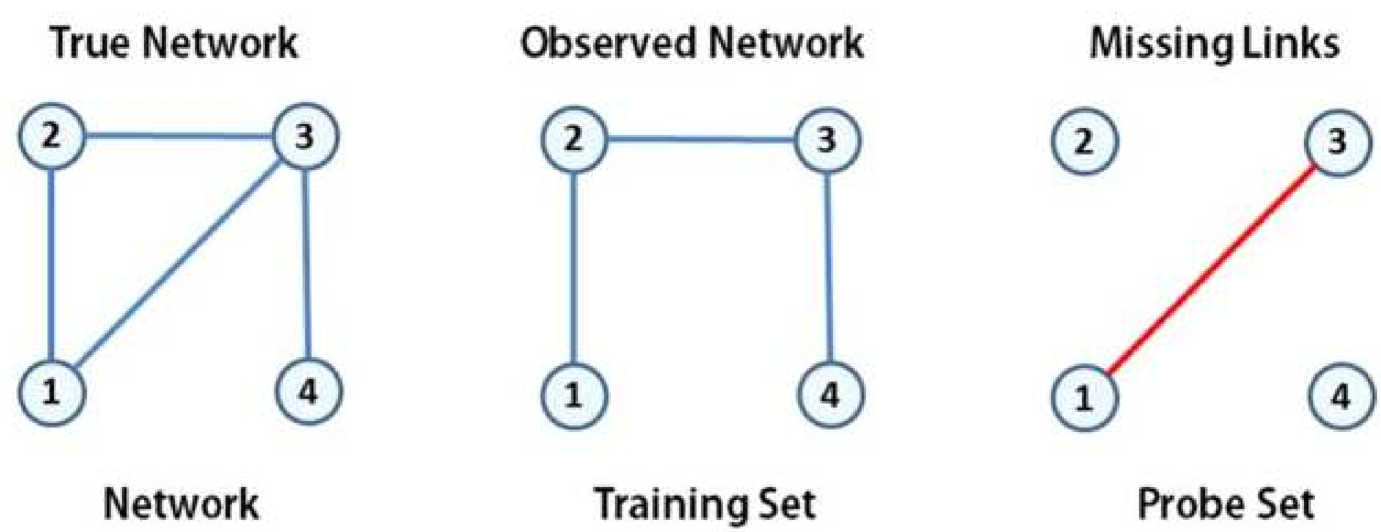
Link prediction

- Link prediction problem:
  - Given the links in the graph at time , predict the edges that will be added to the graph between and time ' in the future.



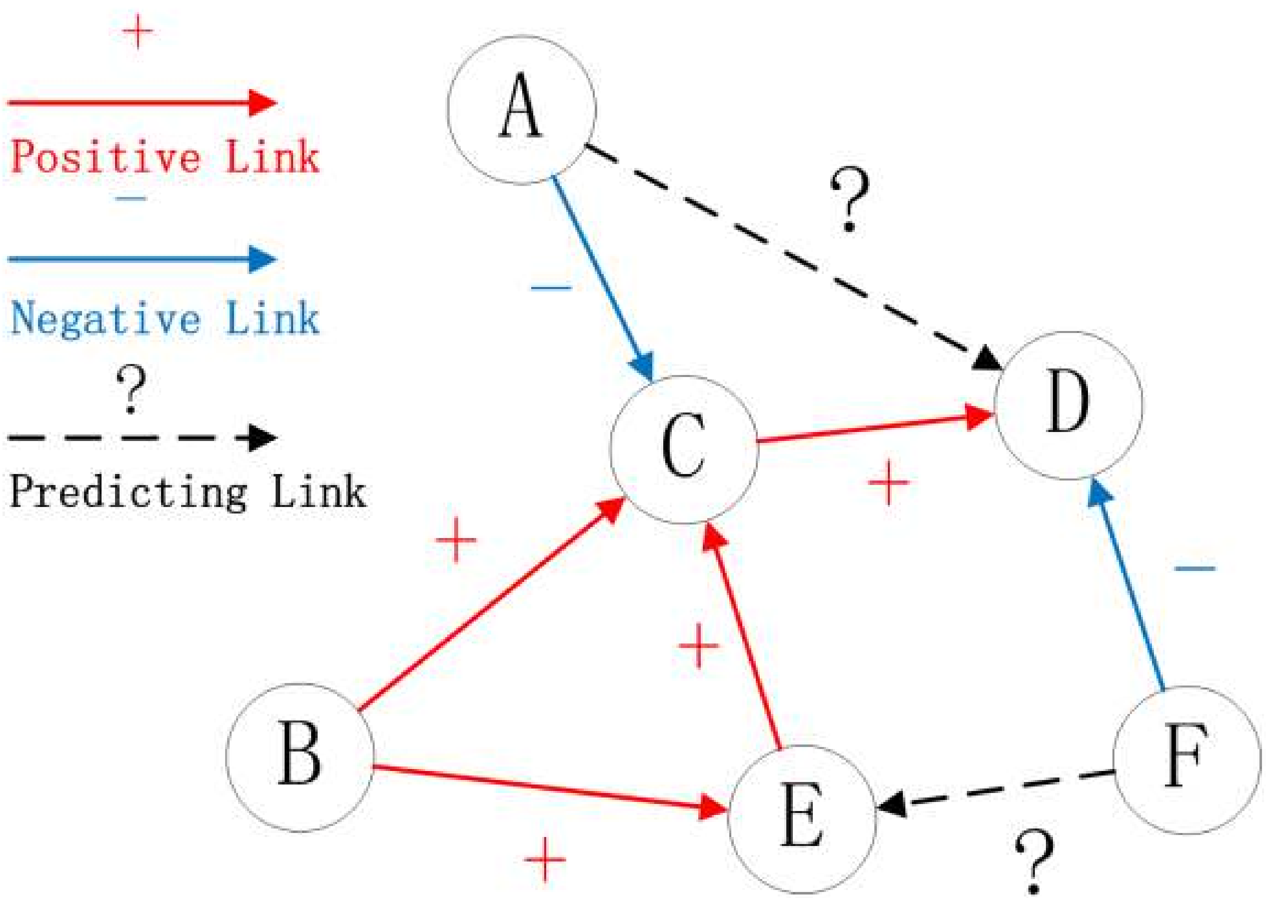
Link inference is missing

- Association prediction differs from missing (hidden) association inference in that:
  - Link prediction to find links that appear over time
  - Missing association inference to find extra links in static graph.



Suy luận liên kết bị thiếu

Link prediction



Application

- Suggestions for making friends in social networks
- Predicting connections between members of a terrorist organization
- Proposal for cooperation among researchers
- Suggest the article should be referenced to
- Predict the product to be purchased
- Predicting interactions of proteins

Link prediction

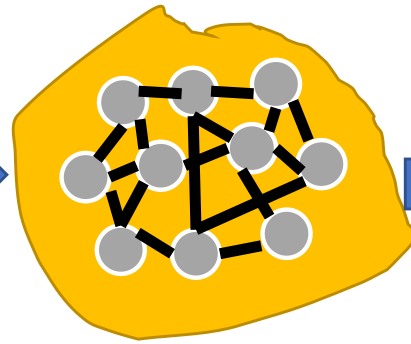
- The link prediction process is to calculate the score $score(u,v)$ for each pair of $<u,v>$, then rank it to choose the v with the highest measure to connect with u.
- Usually divided into three groups:
  - Estimation method based on neighbor vertex: number of common neighbors, Jaccard coefficient, Adamic-Adar, path, ...
  - Methods based on all paths: PageRank, SimRank
  - From another method: bigram, clustering

General pattern

- Step 1: calculate the vertex distance based on some measurement methods such as Jaccard, shortest path, ...
- Step 2: choose a number of pairs of vertices with the closest distance
- Step 3: predict new edge from selected pairs
- Step 4: evaluate the predicted graph with the original graph

General pattern

Generative model



Link Prediction Heuristics

Most likely neighbor of node ***i*** ?

node ***a***

node ***b***

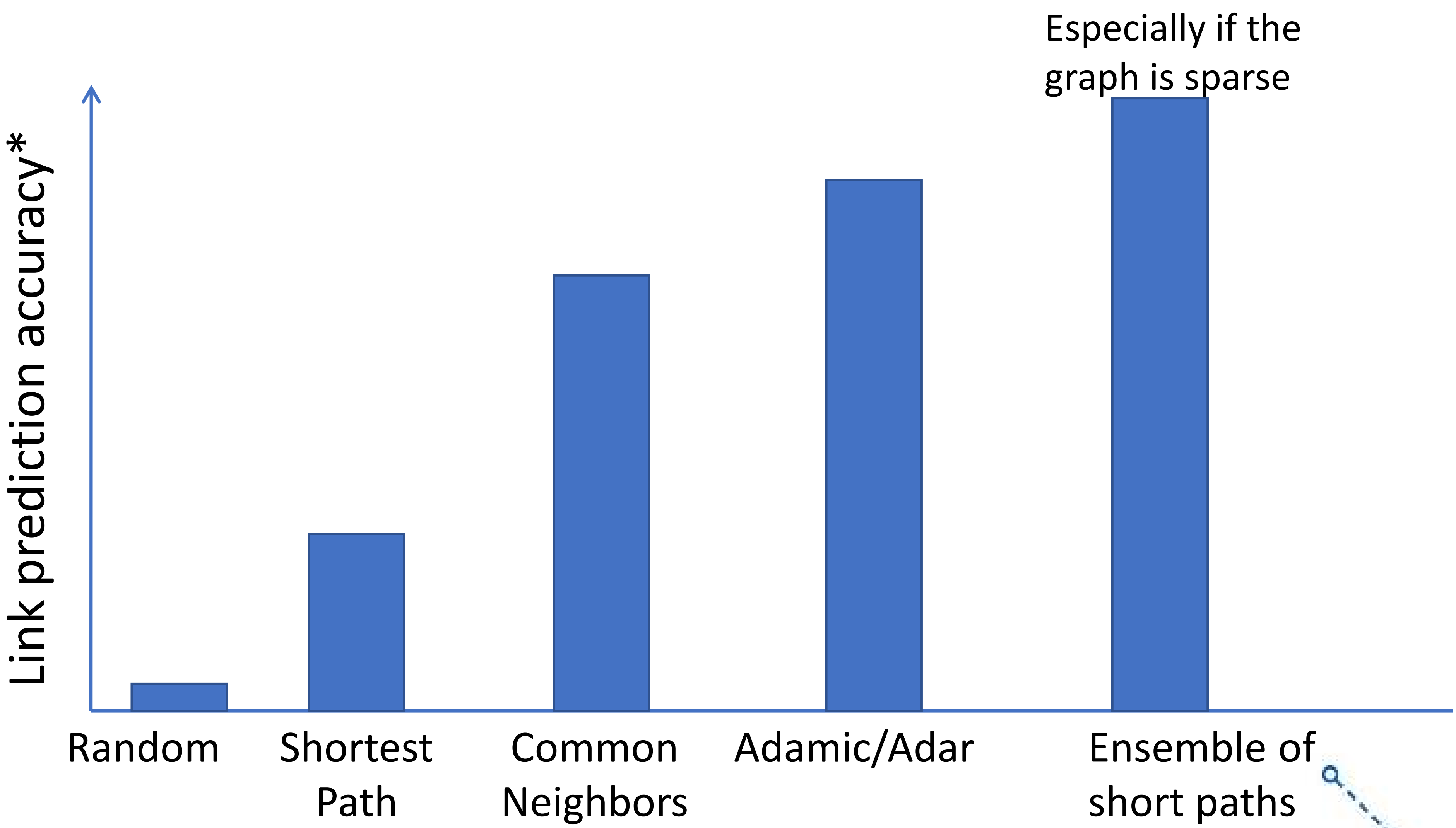
Compare

➔ Can justify the empirical observations

Link prediction

graph distance	(negated) length of shortest path between x and y
common neighbors	$\lvert\Gamma(x) \cap \Gamma(y)\rvert$
Jaccard's coefficient	$\frac{\lvert\Gamma(x)\cap\Gamma(y)\rvert}{\lvert\Gamma(x)\cup\Gamma(y)\rvert}$
Adamic/Adar	$\sum_{z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\log\lvert\Gamma(z)\rvert}$
preferential attachment	$\lvert\Gamma(x)\rvert \cdot \lvert\Gamma(y)\rvert$
Katz$_\beta$	$\sum_{\ell=1}^{\infty} \beta^{\ell} \cdot \lvert \mathsf{paths}_{x,y}^{\langle \ell \rangle} \rvert$ where $\mathsf{paths}_{x,y}^{\langle \ell \rangle} :=$ {paths of length exactly ℓ from x to y} weighted: $\mathsf{paths}_{x,y}^{\langle 1 \rangle} :=$ number of collaborations between x, y. unweighted: $\mathsf{paths}_{x,y}^{\langle 1 \rangle} := 1$ iff x and y collaborate.
hitting time	$-H_{x,y}$
stationary-normed	$-H_{x,y} \cdot \pi_y$
commute time	$-(H_{x,y} + H_{y,x})$
stationary-normed	$-(H_{x,y} \cdot \pi_y + H_{y,x} \cdot \pi_x)$ where $H_{x,y}$:= expected time for random walk from x to reach y π_y := stationary distribution weight of y (proportion of time the random walk is at node y)
rooted PageRank$_\alpha$	stationary distribution weight of y under the following random walk: with probability α, jump to x. with probability $1-\alpha$, go to random neighbor of current node.
SimRank$_\gamma$	$\begin{cases} 1 & \text{if } x = y \\ \gamma \cdot \frac{\sum_{a\in\Gamma(x)} \sum_{b\in\Gamma(y)} \mathsf{score}(a,b)}{\lvert\Gamma(x)\rvert\cdot\lvert\Gamma(y)\rvert} & \text{otherwise} \end{cases}$

Comparison of methods



***Liben-Nowell & Kleinberg, 2003; Brand, 2005; Sarkar & Moore, 2007**

Content

- Link prediction
- Learn in link prediction
  - Unsupervised learning
  - Supervised learning

Learn in link prediction

- Learning in link prediction is divided into two types:
  - Unsupervised: No train test
    - Similarity-based: similar vertices are connected
    - Cluster-based: vertices from the same group demonstrate similar connectivity patterns.
  - Supervised: provide a set of associated vertices to train the model

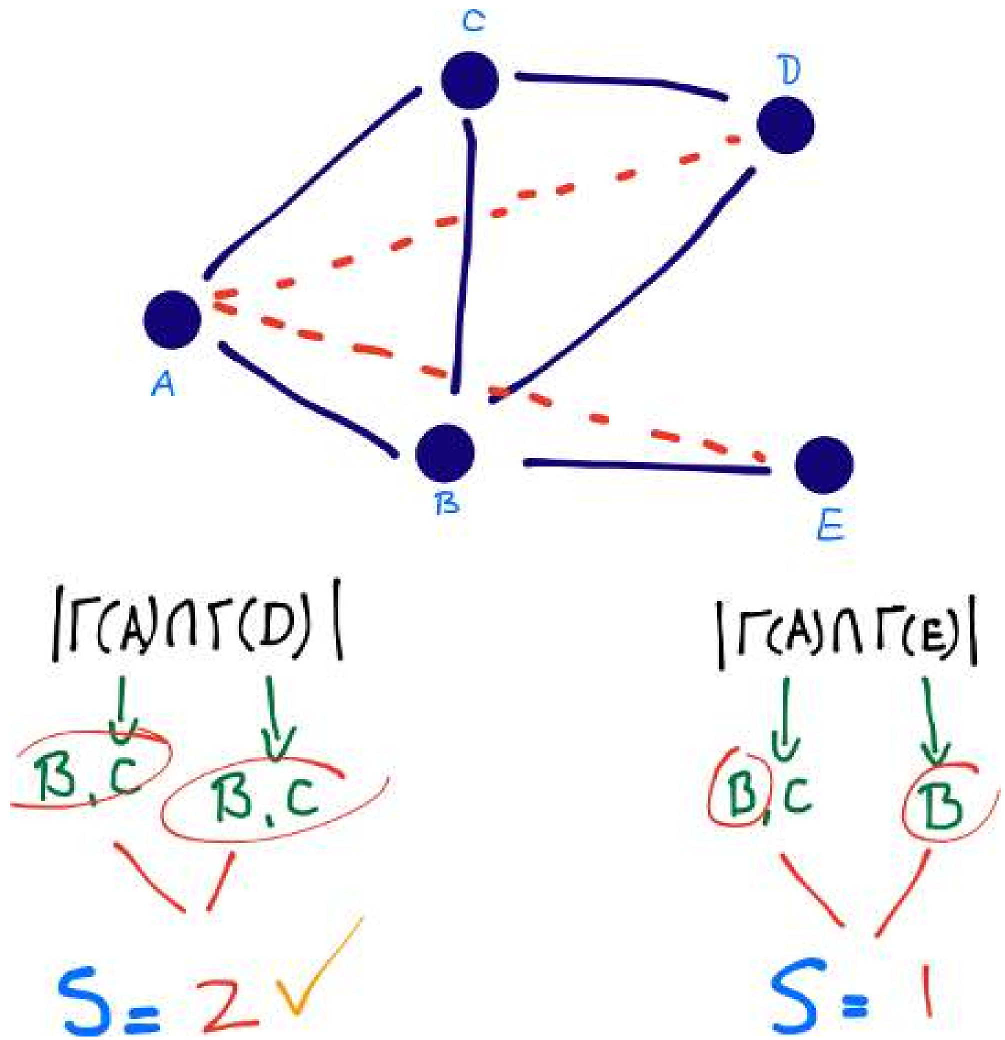
Link prediction based on similarity

- Similarity (unsupervised learning) measures the distance between vertices in a graph.
- Classify:
  - 1-step: neighboring vertices are connected
  - Laplacian: dissimilar vertices are connected (L = D-A)
  - Degree: vertices of similar degree
  - A^2: vertices shared with neighbors(2-hop)
  - Closeness: vertices have nearly the same center
  - Betweeness: vertices have the same intermediate center

Based on neighboring vertices

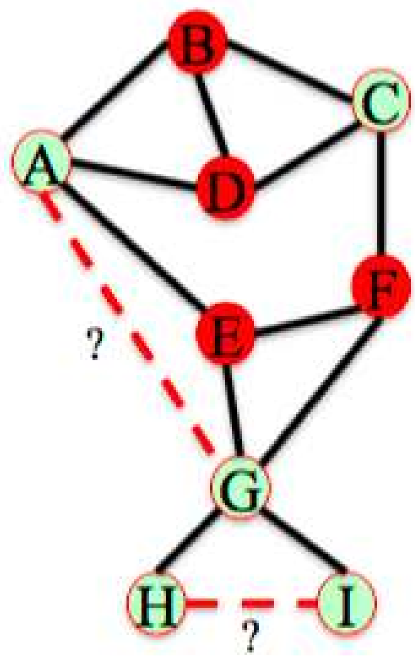
- Common neighbors proposed by Ahmad Sadrei as the simplest measure.
  - Considered the effect of triangular closure (closing a triangle)
- The first is to find the intersection between two neighbors of two vertices, the measure of similarity between the two vertices is the number of elements in this set.
- $score(u, v) = |N(u) \cap N(v)|$
- If this measurement is greater than the given threshold, a link is created

Based on neighboring vertices



Jaccard coefficient

- The Jaccard coefficient normalizes the number of common neighbors by the total number of neighbors.
- $score(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$



$$\text{jacc\_coeff}(A, C) = \frac{|\{B, D\}|}{|\{B, D, E, F\}|} = \frac{2}{4} = \frac{1}{2}$$

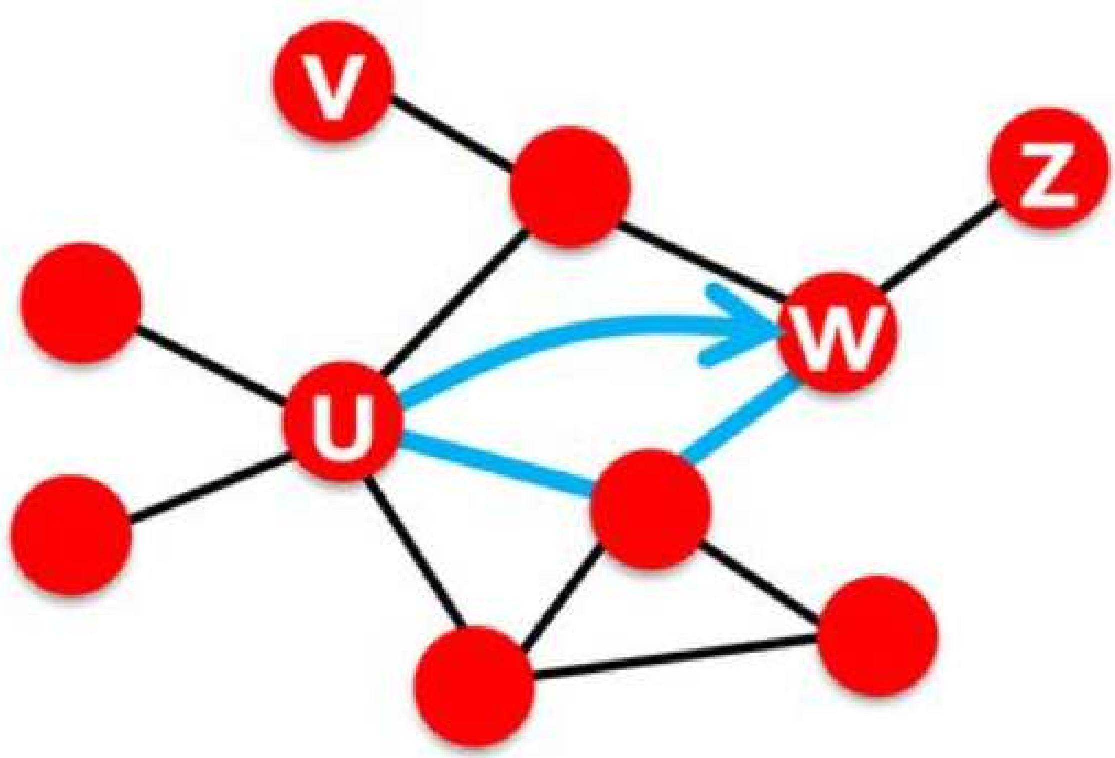
Adamic Adar measure

- Measure proposed by Lada Adamic and Eytan Adar (2003)
- In addition to counting the number of common neighbors, the Adamic Adar measure also sums the log inverses of the neighbors' degrees.
  - Counts common neighbors but lowers vertices that have too many neighbors
  - The triangle closure effect is heavily influenced by low-order vertices.

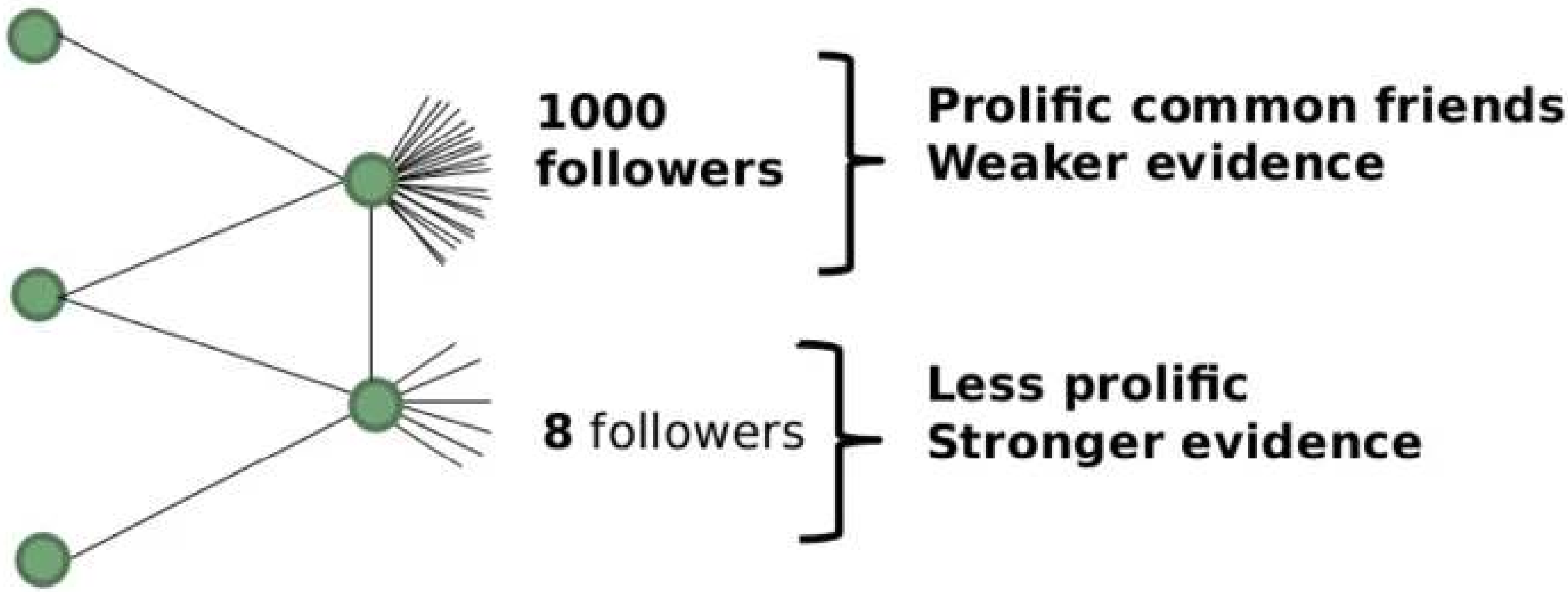
$$score(u, v) = \sum_{z \in N(u) \cap N(v)} \frac{1}{\log(N(z))}$$

Triangle closing effect

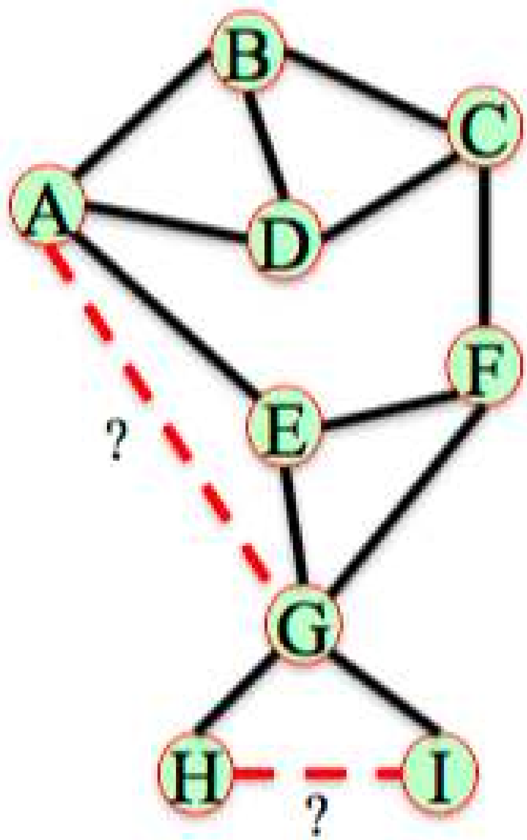
- 92% of new Facebook friend connections are in friend-of-a-friend



Adamic Adar measure



Adamic Adar measure

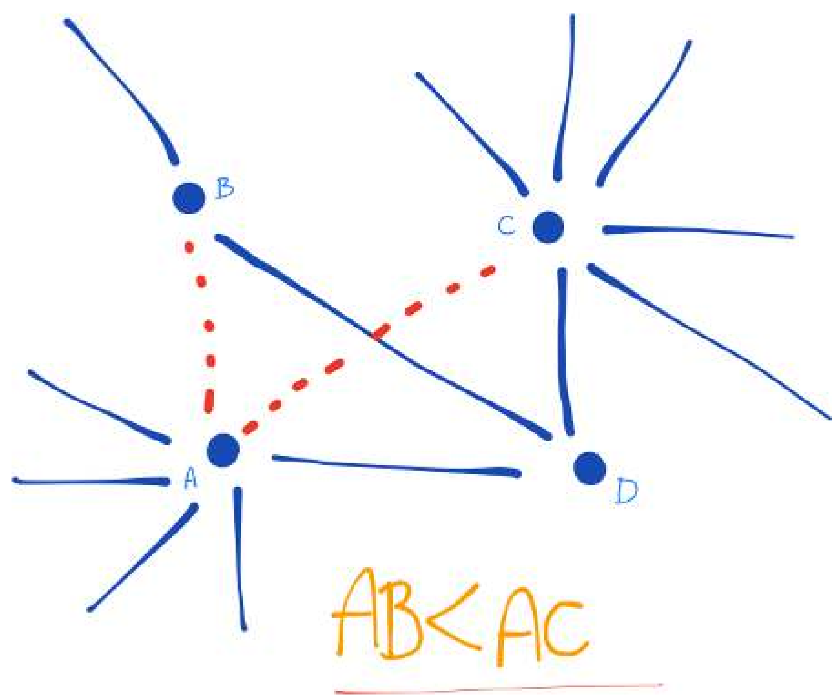


$$\text{adamic\_adar}(A, C) = \frac{1}{\log(3)} + \frac{1}{\log(3)} = 1.82$$

Preferential attachement

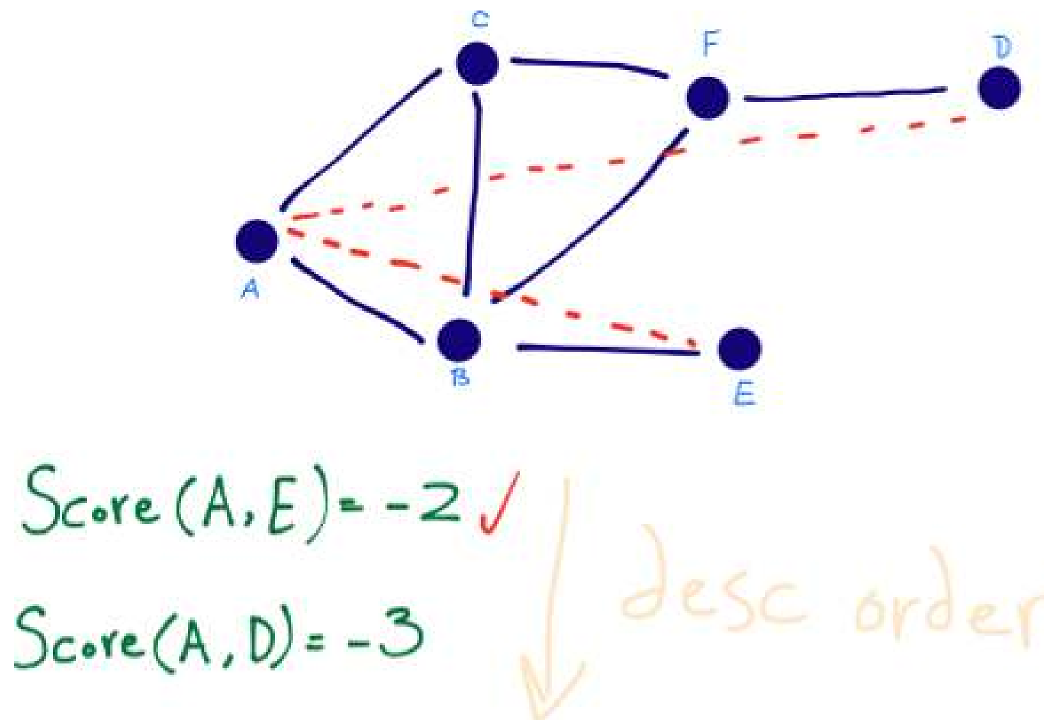
- Preferential attachement proposed by Albert-László Barabási and Réka Albert to describe the phenomenon of vertices with many relationships tending to be interconnected.
- “Rich-get-Richer”

$$score(u,v) = degree(u) \times degree(v)$$



Shortest path

- The pair of vertices with the shortest path will tend to connect
- $score(u, v) = -shortestPath(u, v)$



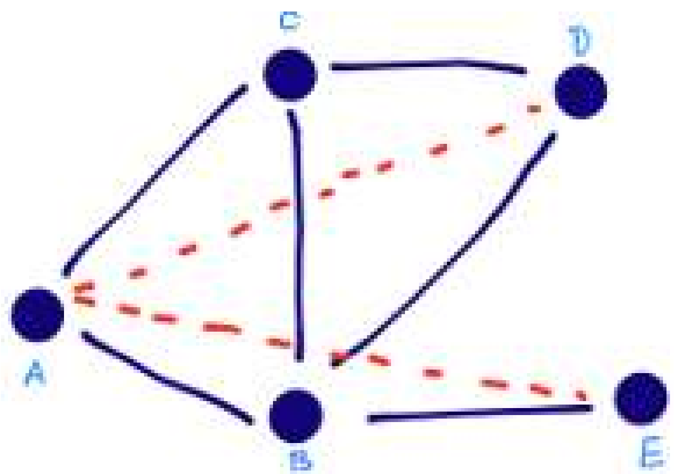
Katz measure

- The Katz measure considers not only the shortest path between vertices, but also all paths between them.
- The shorter the path, the more weighted

$$score(u, v) = \sum_{l=1}^{\infty} \beta^{l} |paths_{u,v}^{<l>}|$$

with $paths_{u,v}^{<l>}$ is the set of length paths l between you and v, β is a very small constant to make the path as long as possible with little contribution to the sum.

Katz measure



\# of Hops

$Path^{2}_{A,D} = 2$ $\quad$ $Path^{3}_{A,D} = 2$

$S = \frac{1}{2} \cdot 2 + \frac{1}{4} \cdot 2 + \cdots$

Damping Factor

$Path^{2}_{A,E} = 1$ $\quad$ $Path^{3}_{A,E} = 1$

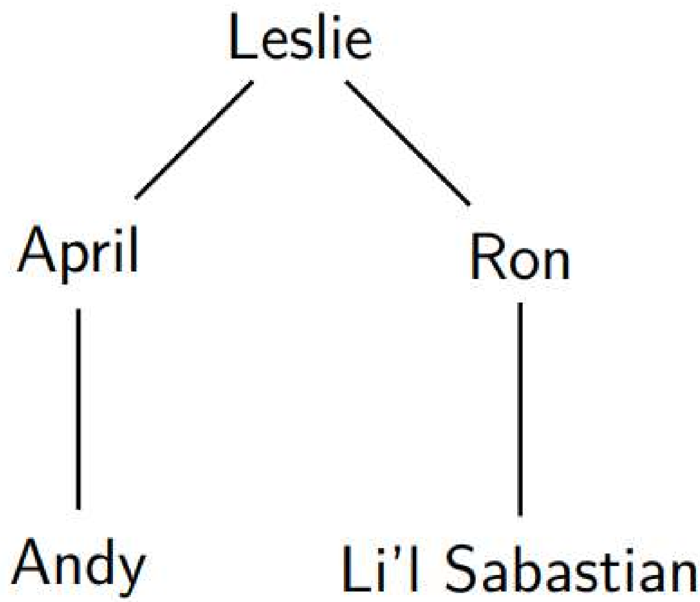
$S = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 1 + \cdots$

SimRank

- SimRank Calculated based on the measure of neighbors, meaning that the more similar the neighbors, the more connected those two vertices tend to be.
- $score(u, v) = \frac{C}{|N(u)|.|N(v)|} \sum_{z \in N(u)} \sum_{z' \in N(v)} score(z, z')$

with C is the constant in paragraph [0,1]

SimRank



$$
\begin{aligned}
s(April, Ron) &= 0.4 \\
&\downarrow \\
s(Andy, Li'lSebastian) &= 0.33
\end{aligned}
$$

Hitting Time

- The hitting time from vertex u to v is the random number of steps expected to meet vertex v when starting from u.
- Similar measurement based on hitting time:

  $score(u, v) = -H_{u,v}$
- However, some vertices with large connections will easily lead to an immediate random walk to it no matter where it comes from. To avoid this phenomenon, we standardize it: $score(u, v) = -H_{u,v}\pi_v$

Commute time

- Because the hitting time is asymmetrical, we can calculate: $score(u,v) = -(H_{u,v}\pi_v + H_{v,u}\pi_u)$

Closeness centrality

- Closeness centrality: An agent I is close center if it can interact easily with all other agents.
- Or, the distance of i to all other agents is short

Closeness centrality (tt)

- *The shortest distance from agent i to agent j (symbol d(i,j)) is measured by the number of links on the shortest path.*
- The closeness centre of agent i is denoted Cc(i) and is normalized with n-1 as the sum of the shortest distances from i to all other agents.

Closeness centrality (tt)

- For a scalar graph: the center near Cc(i) of agent i is defined as:

$$C_c(i) = \frac{n-1}{\sum_{j=1}^{n} d(i,j)}$$

Note: this expression is only possible in the case of a connected graph

Betweenness centrality

- Betweenness centrality: If two nonadjacent agents j and k want to interact and agent i is between j and k, then i may have some control over their interactions.
- If i is in the path of many of these kinds of interactions, then i is an important agent.

Betweenness centrality (tt)

- For a scalar graph, the intermediate properties of an agent i are defined by the number of shortest paths through i (symbols pjk(i), j ≠ i and k ≠ i) and normalized by the total number of shortest paths of all agent pairs except i:

$$C_B(i) = \sum_{j<k} \frac{p_{jk}(i)}{p_{jk}}$$

Betweenness centrality (tt)

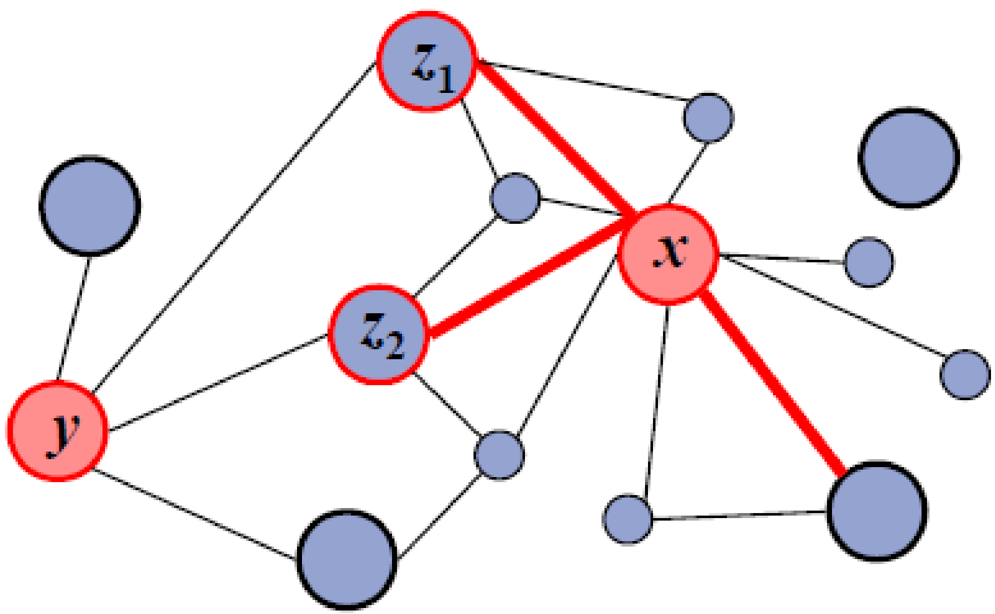
- To ensure that the value is between 0 and 1, CB(i) is normalized with (n-1)(n-2)/2, which is the maximum value of CB(i):

$$C_B(i) = \frac{2\sum_{j<k} \frac{p_{jk}(i)}{p_{jk}}}{(n-1)(n-2)}$$

- Unlike proximity, intermediate can be calculated even if the graph is not connected.

Unseen Bigram

- Bigram (N-gram) Expresses two characters/words standing next to each other in a natural language sentence.
- If the bigram does not appear in the training set but appears in the test set, it is called unseen bigram.
- Use a method in the language model to calculate.



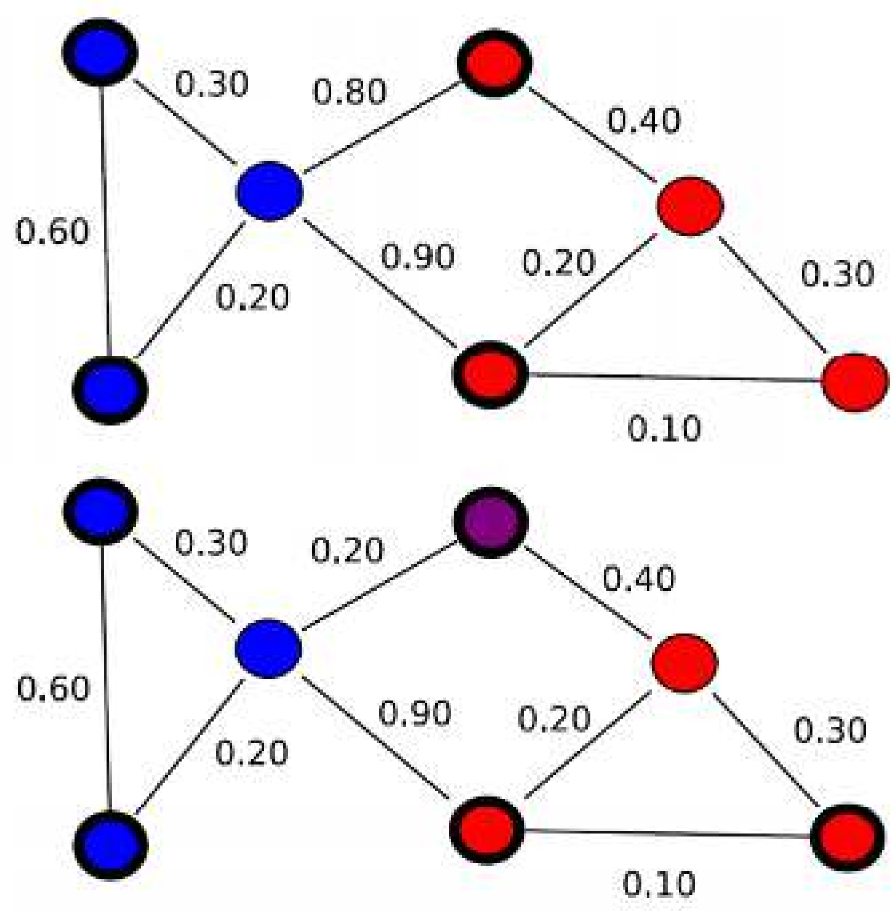
Unseen Bigram

- Put $S_u^{<\delta>}$ is a vertex δ set that is highly similar to you through some measure:
- $score_{unweighted}(u,v) = \left|\left\{z: z \in N(v) \cap S_u^{<\delta>}\right\}\right|$

$$score_{weighted}(u,v) = \sum_{z \in N(v) \cap S_u^{<\delta>}} score(u,z)$$

Based on grouping

- Apply several grouping methods (e.g. DB-Scan) to perform graph grouping (community detection).
- Connect vertices in a group based on defined criteria.



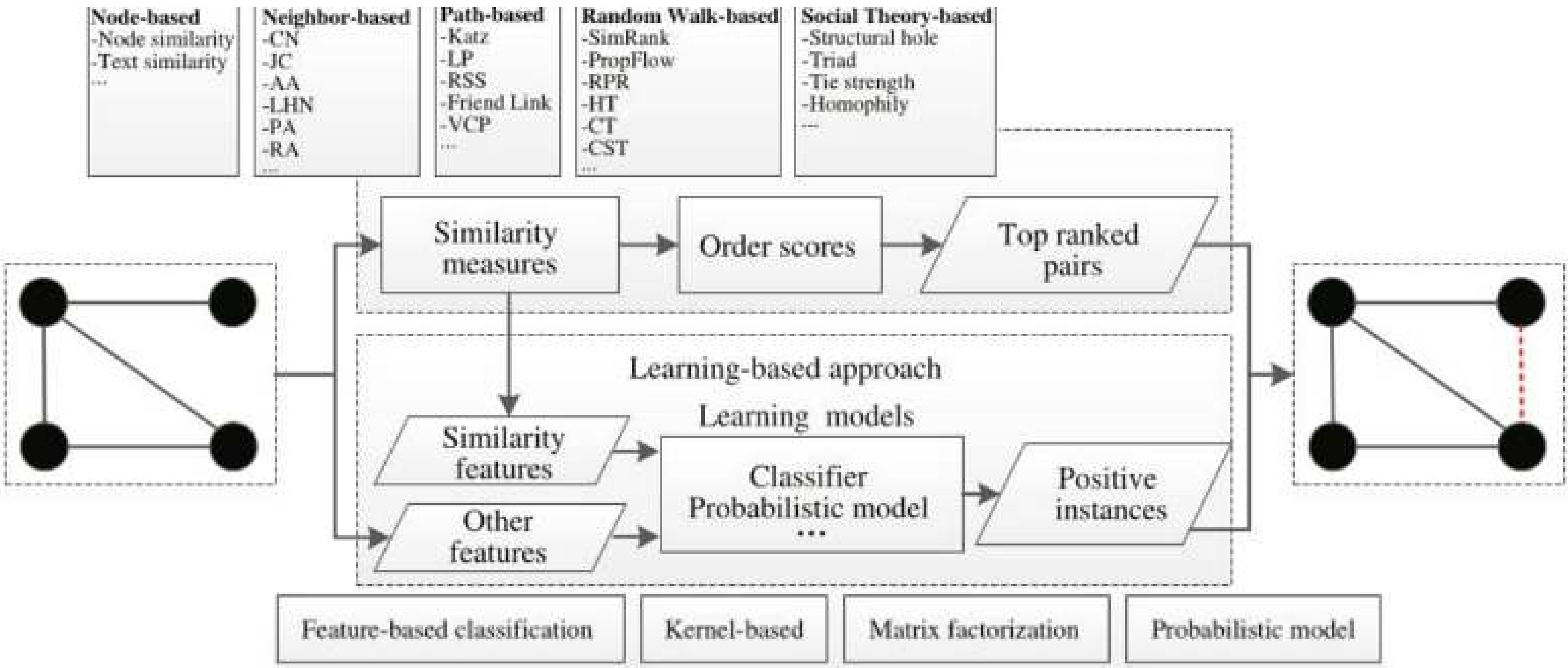
Unsupervised Method Review

- After an analogue measure is selected, the result is a list of the most similar pairs filtered out.
- Precision, recall, or accuracy can be evaluated using a test graph.
- However, the quality is often very low when used in a real-world network due to multiple edges being created for no logical reason.

Content

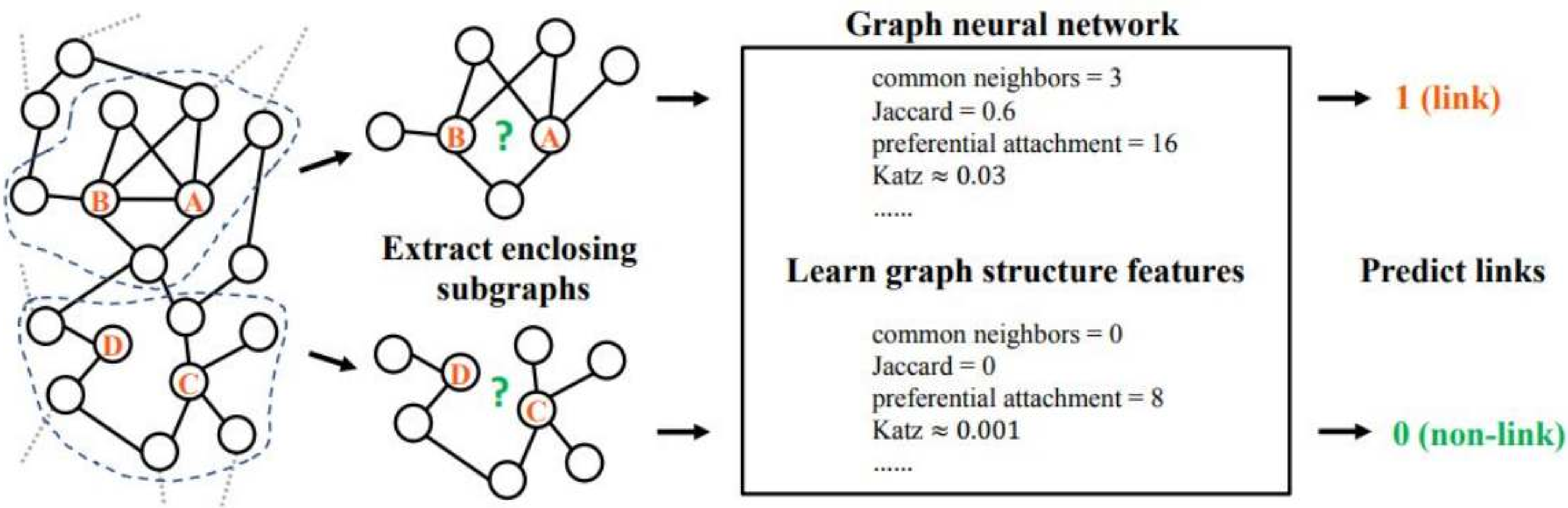
- Link prediction
- Learn in link prediction
  - Unsupervised learning
  - Supervised learning

Learn to predict



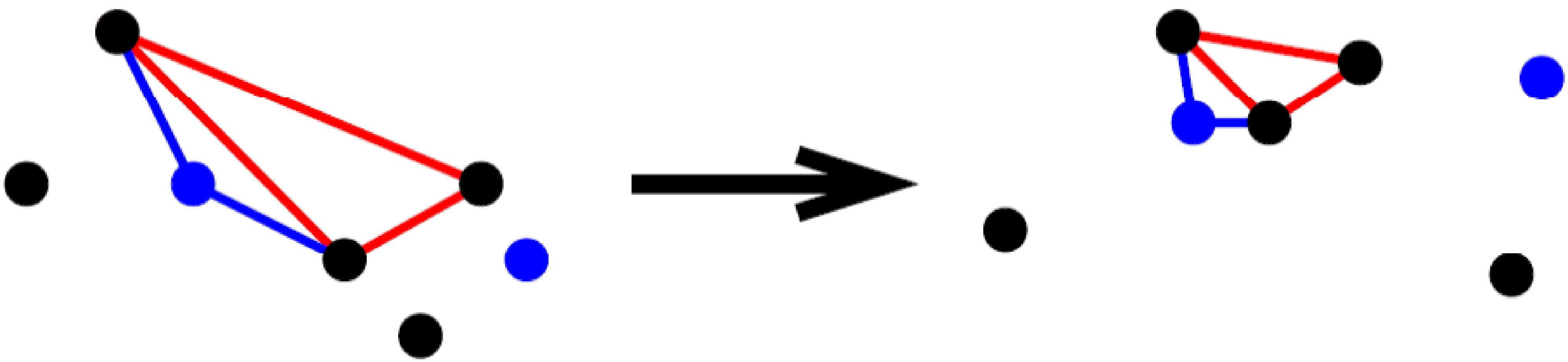
Learning features

- Local cartouche-based learning



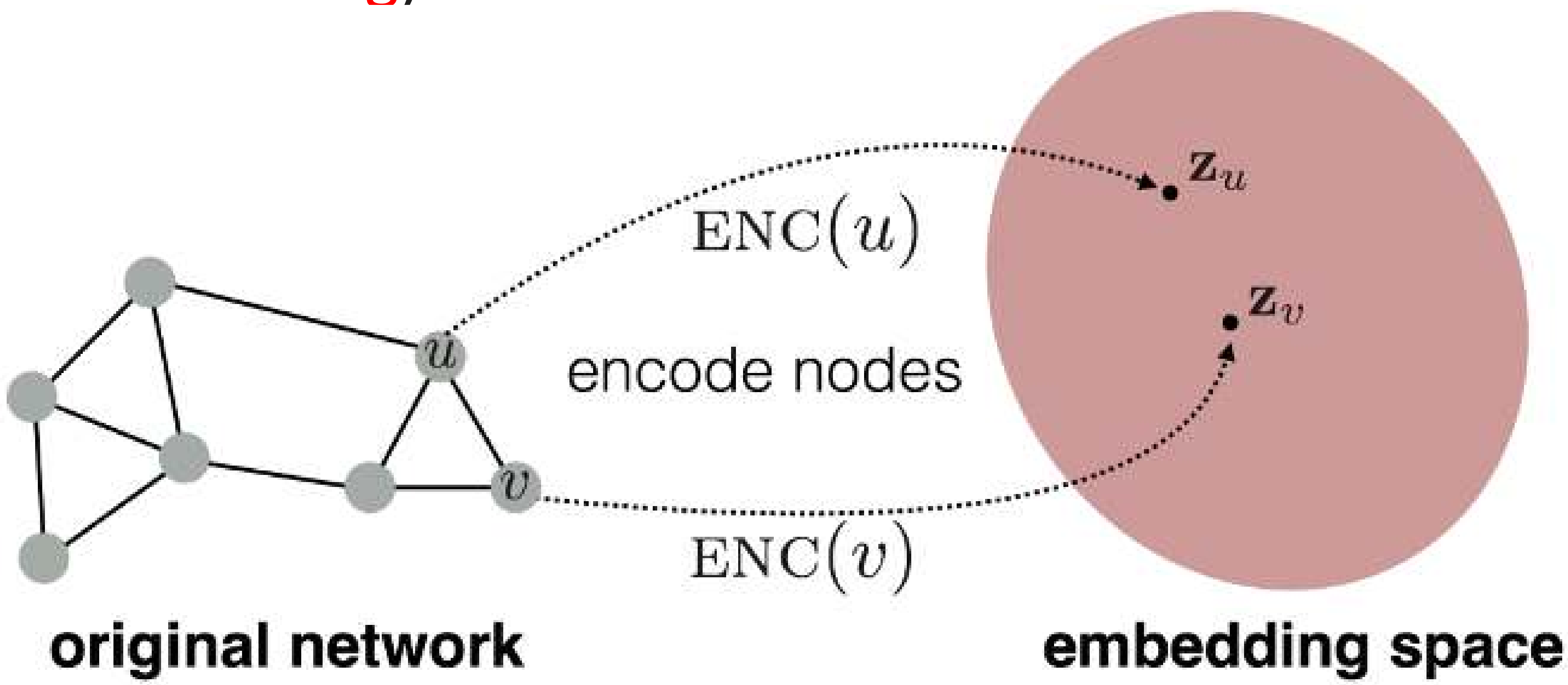
Learning on global characteristics

- Learning on global characteristics
  - Use a known subnet to readjust the distance before applying similarity



Embedded-based learning

- It is possible to learn based on conversions to other dimensions to determine similarity (vertex embedding).



G -> embedded in vector space -> calculate distance

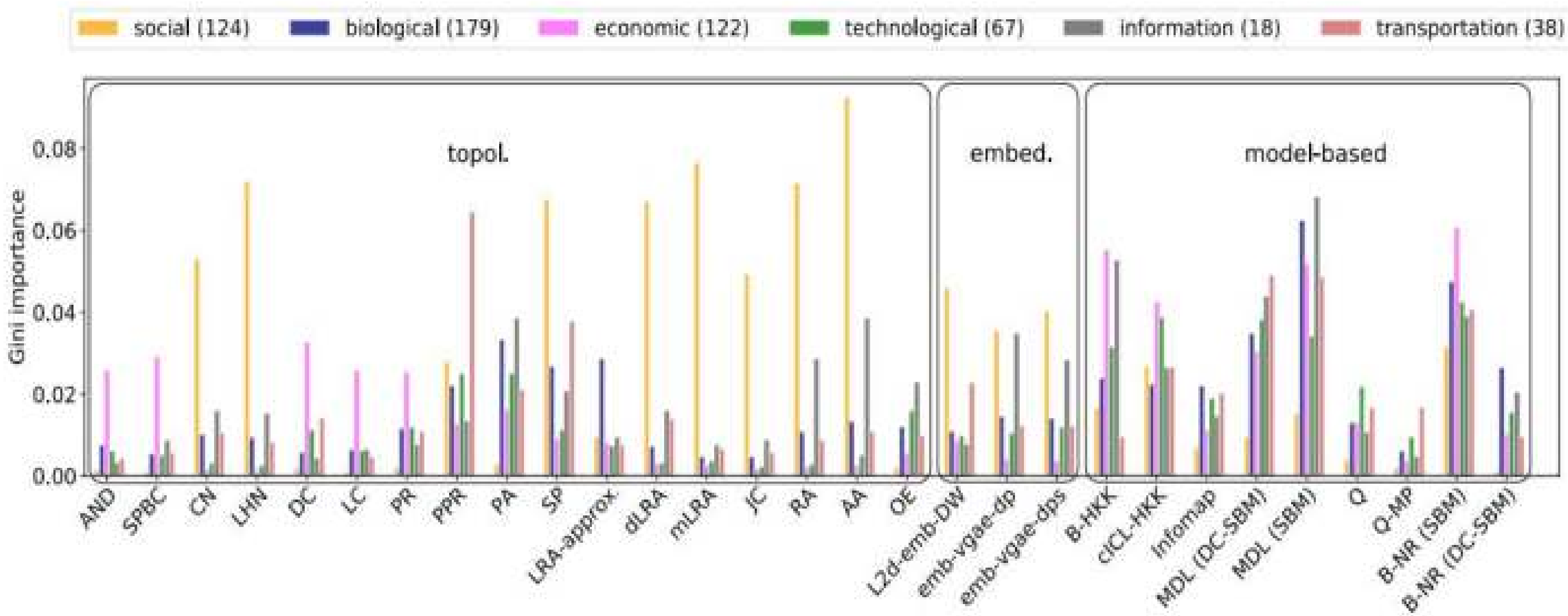
Target function

$$F(\mathbf{A}_{source}) \backsimeq \mathbf{A}_{target}$$

$$\min ||F(\mathbf{A}_{source}) - \mathbf{A}_{target}||_{\mathrm{F}}$$

Which is best?

550 structurally diverse networks from six scientific domains



no one predictor or family is best, or worst, across all realistic inputs

References

- http://redalertproject.eu/wp-content/uploads/2019/05/D3-2-Link-prediction-models-FINAL.pdf