Collective Entity Resolution in Relational Data

CompSci 590.03
Instructor: Ashwin Machanavajjhala

Recap: Constraints

- **Transitivity**:
  If x and y match, y and z match, then x and z must match
  - Useful in deduplication

- **Exclusivity**:
  If x matches with y, then z cannot match with y
  - Useful in record linkage (matches across two datasets)
  - Each dataset does not have any duplicates.

- **Relational Constraints**:
  If x and y match, then z and w should match
  - If movies are the same, then directors must be the same
  - (We will see in next class)

Recap: Constraint Types

	Hard Constraint	**Soft Constraint**
Positive Evidence	Transitivity: x=y & y=z => x=z	**Relational: If x, y match then z, w are more likely to match** ***If two venues match, then their papers are more likely to match***
Negative Evidence	Exclusivity: x and y must refer to distinct entities **Relational: If x,y don't match then z,w cannot match** ***If two venues don't match, then their papers don't match***	Soft Exclusivity: x and y are very likely different elements

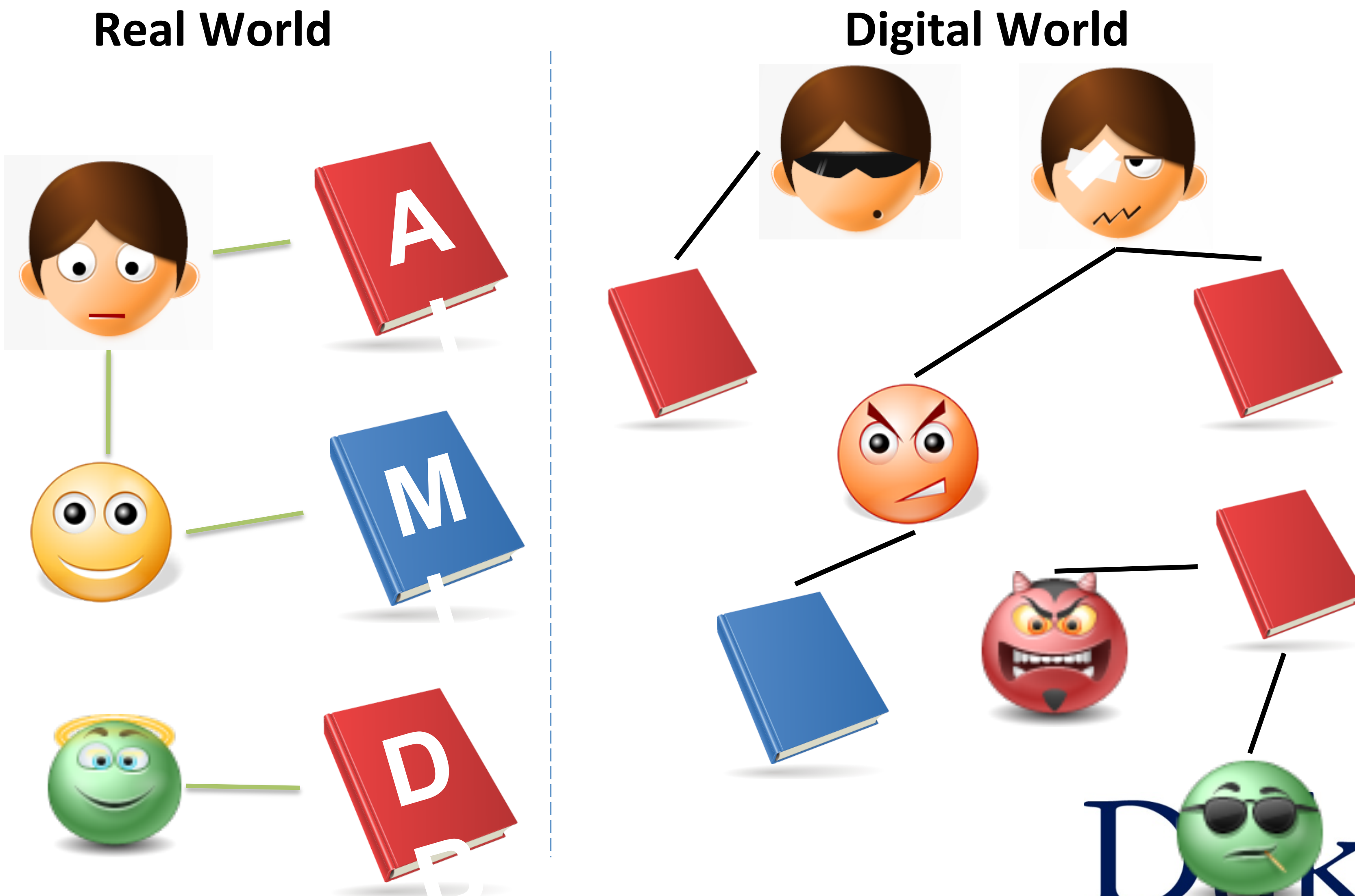
Match Dependencies

When matching decisions depend on other matching decisions (in other words, matching decisions are not made independently for each pair), we refer to the approach as ***collective***

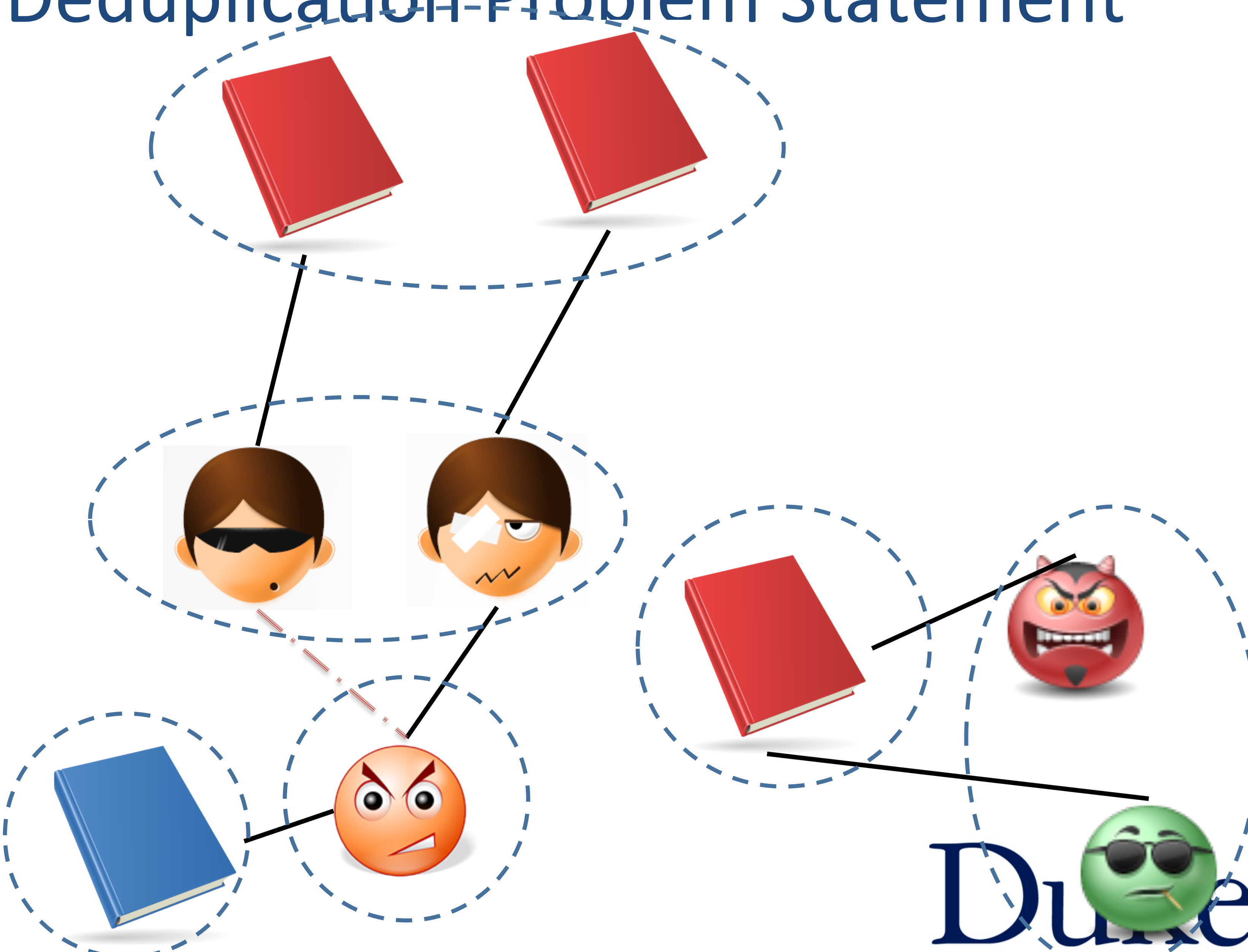
This Class

- Collective Entity Resolution for Relational Data
  - Problem Statement
  - Motivating Example
  - Similarity functions for Linked Data
  - Relational Clustering

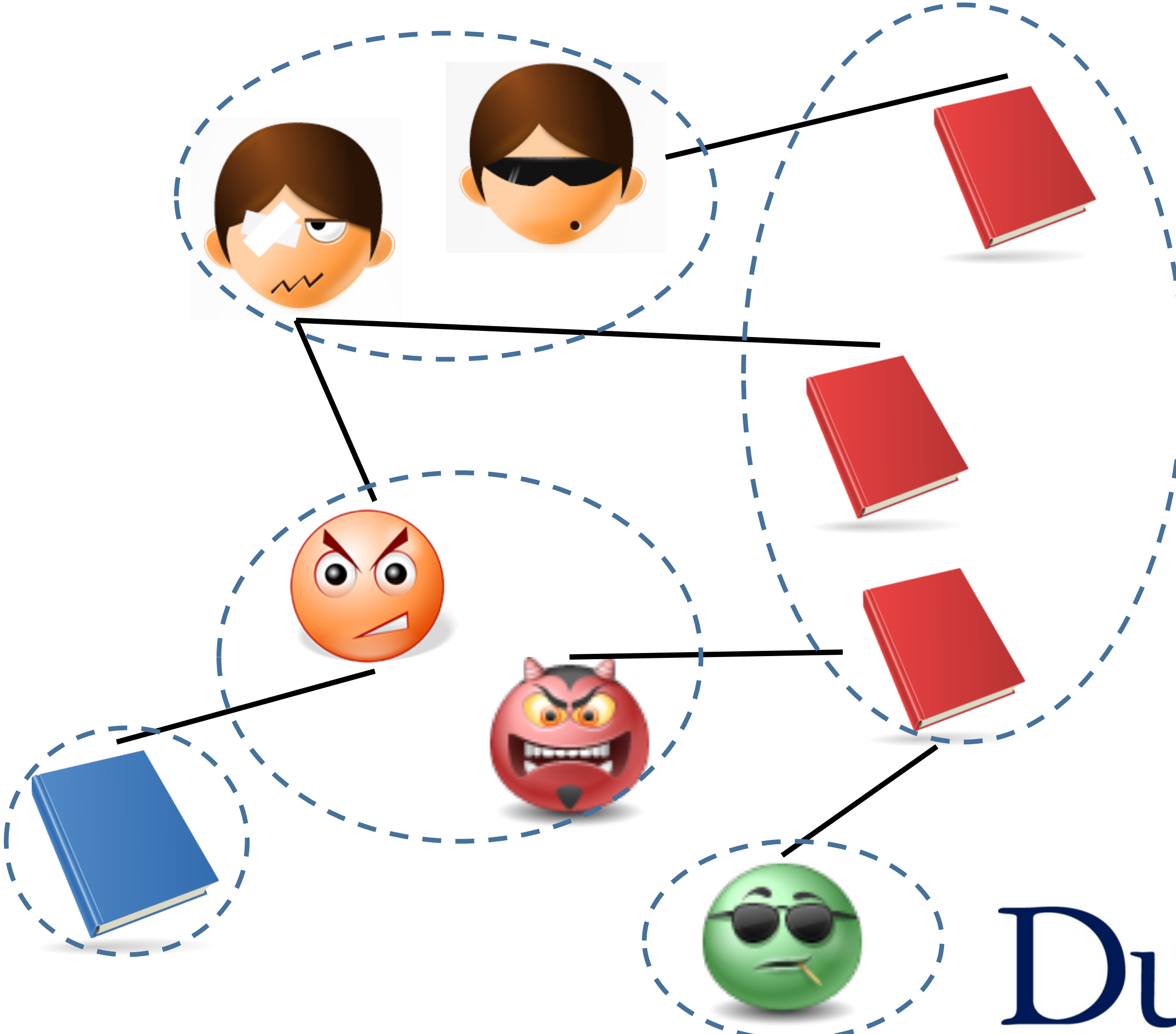
Abstract Problem Statement



Deduplication Problem Statement



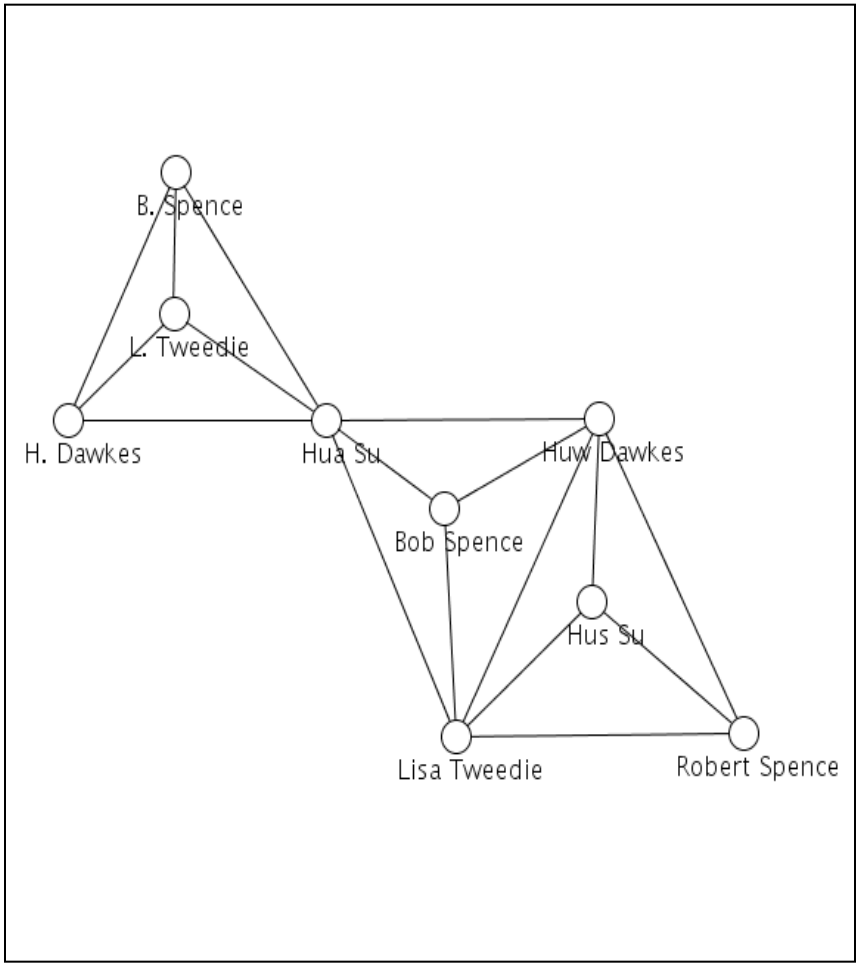
Relationships are crucial



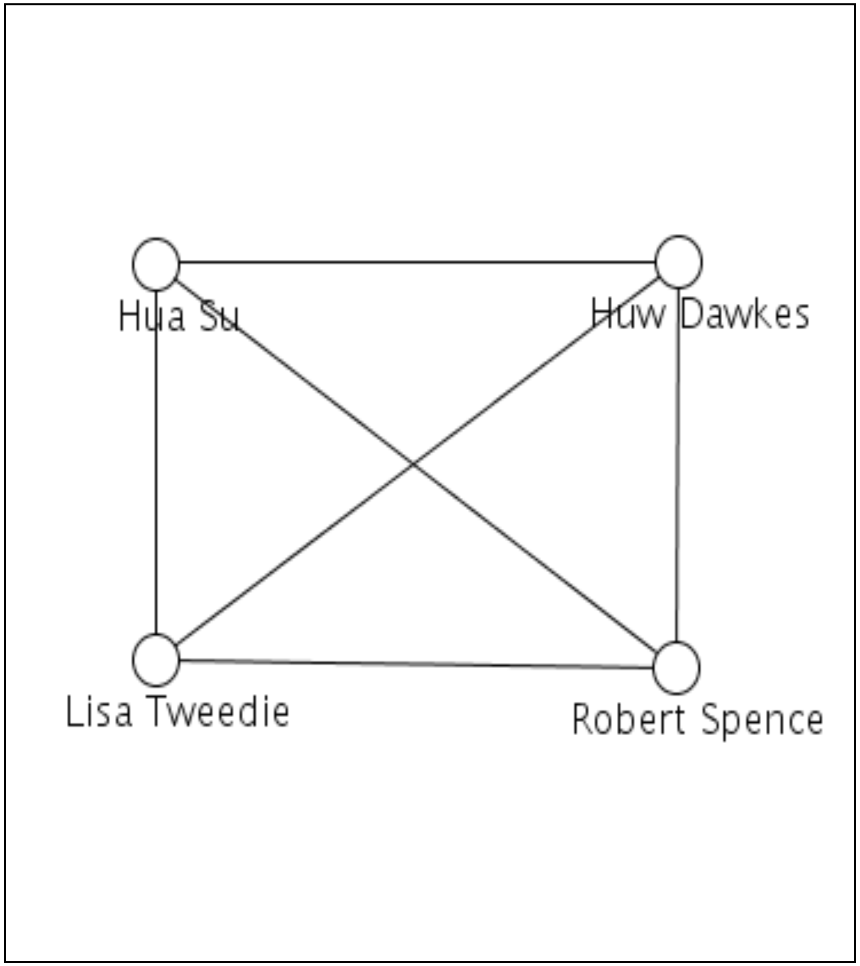
This Class

- Collective Entity Resolution for Relational Data
  - Problem Statement
  - Motivating Example
  - Similarity functions for Linked Data
  - Relational Clustering

InfoVis Co-Author Network Fragment

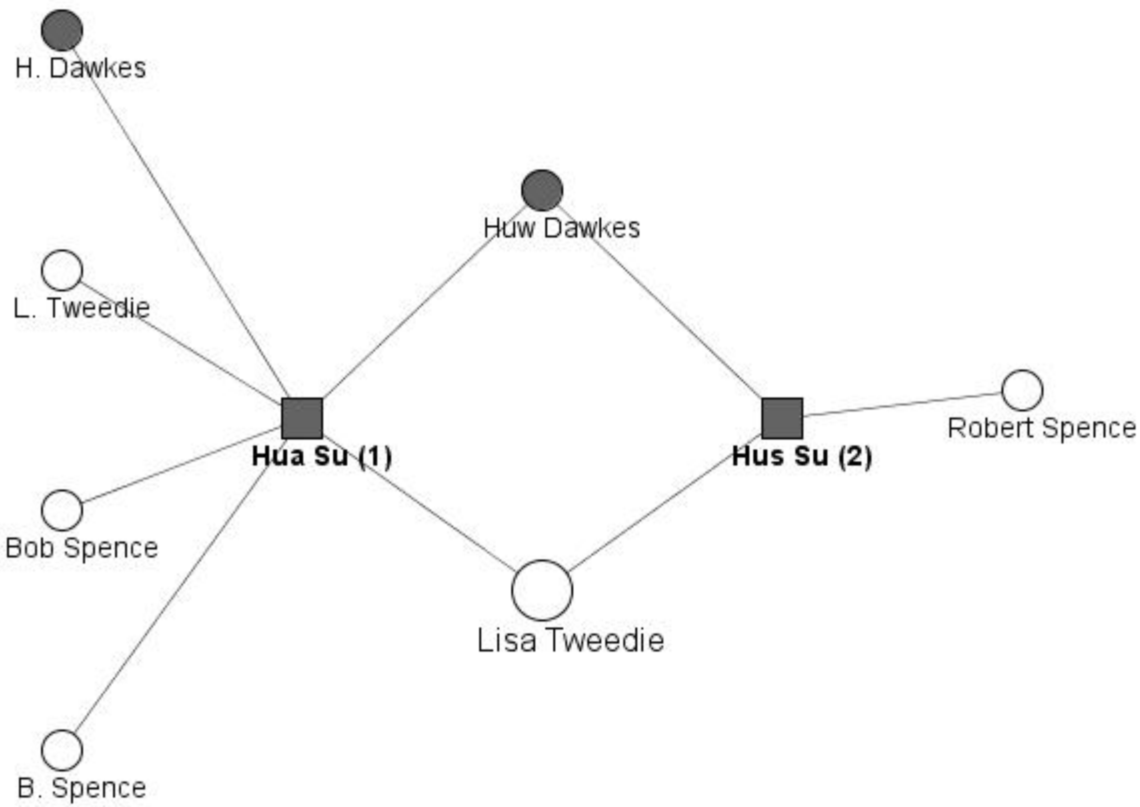


before



after

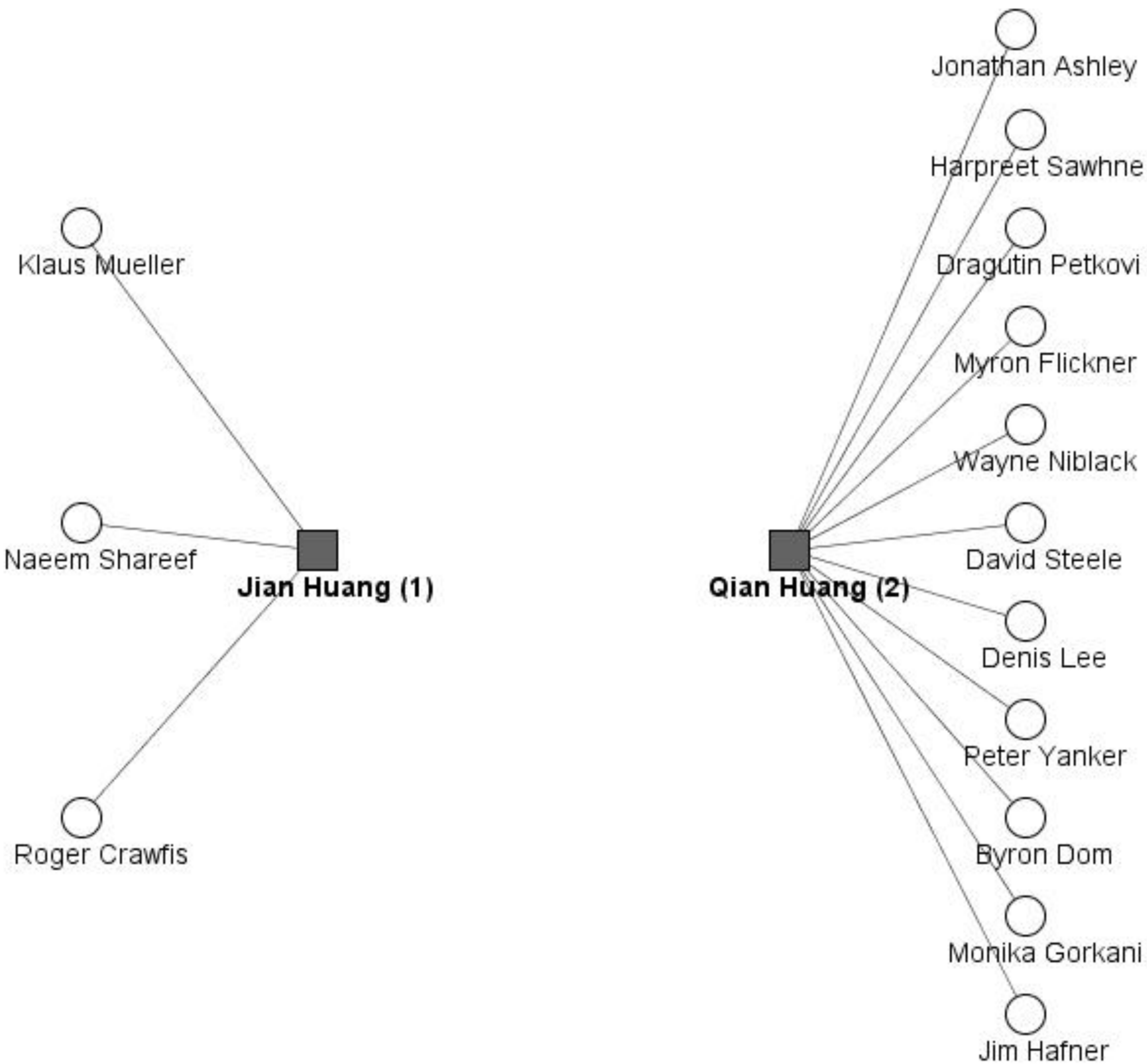
Relational Constraints



Very similar names.

Added evidence from shared co-authors

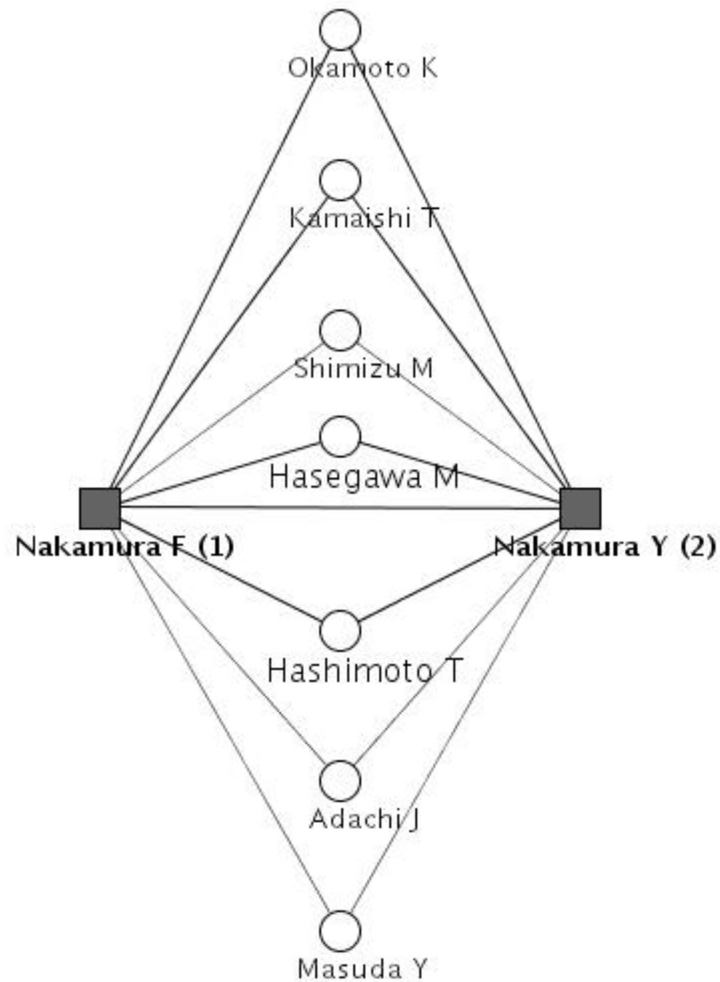
Relational Constraints



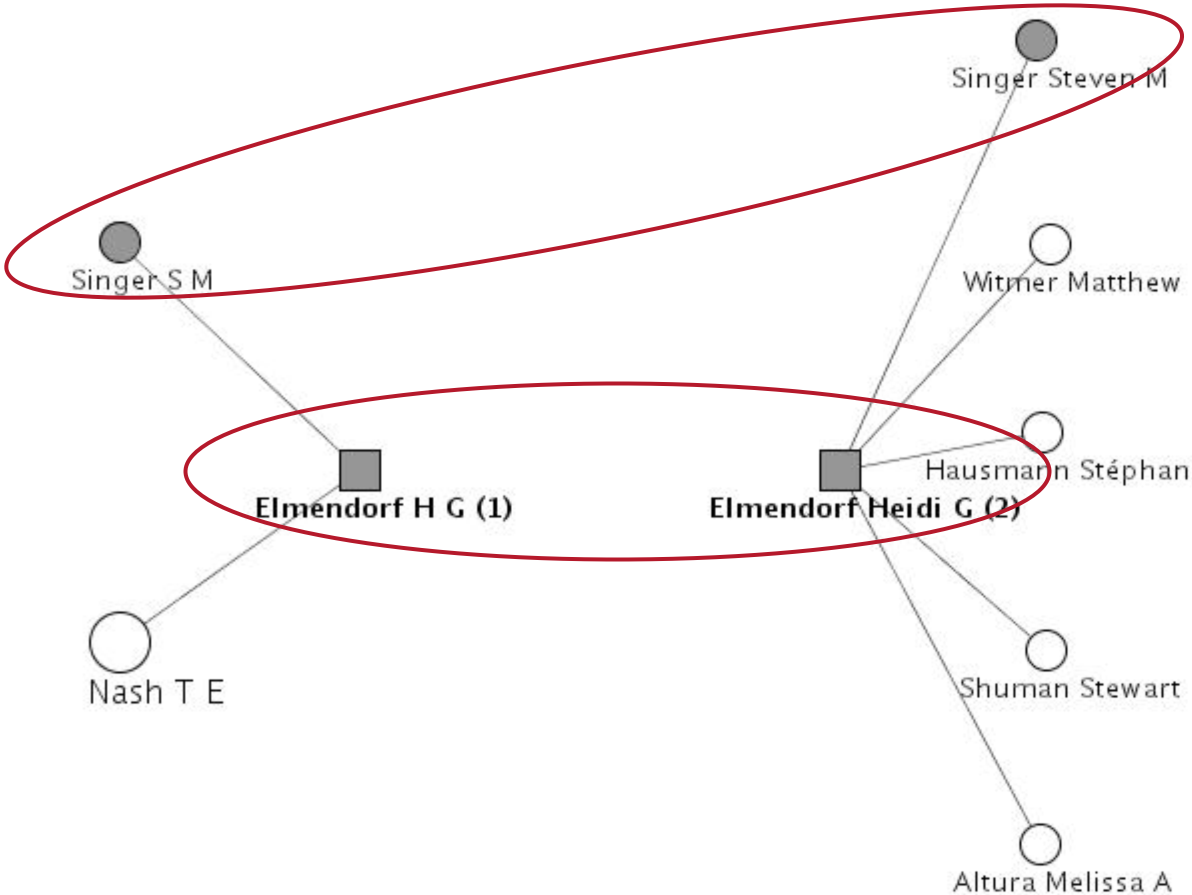
Very similar names but no shared collaborators

Relational Constraints

Co-authors are typically distinct



Collective Entity Resolution



One resolution provides evidence for another => joint resolution

Duke UNIVERSITY

This Class

- Collective Entity Resolution for Relational Data
  - Problem Statement
  - Motivating Example
  - Similarity functions for Linked Data
  - Relational Clustering

Relational Features

- There are a variety of ways of improving ER performance when data is richer than a single table/entity type

- One of the simplest is to use additional information, to *enrich* model with *relational features* that will provide richer context for matching

Examples of relational features

- Value of edge or neighboring attribute (1-1)

- Aggregates (1-many)
  - Mode (sum, min, max) of related attribute

- Set similarity measures to compare nodes based on set of related nodes, e.g., compare neighborhoods
  - Overlap
  - Jaccard coefficient
  - Average similarity between set members

Preferential Attachment Score

[Liben-Nowell & Kleinberg, JASIST07]

- Based on studies, e.g. [Newman, PRL01], showing that people with a larger number of existing relations are more likely to initiate new ones.

$$s(a,b) = |N_a| \cdot |N_b|$$

Set of a's neighbors

Duke UNIVERSITY

Common Neighbors

- Two nodes are likely to be connected in a graph if they share a large number of common neighbors.

$$s(a,b) = \underbrace{N(a) \cap N(b)}_{\text{Can be any kind of shared attributes or relationships to shared entities}}$$

Adamic/Adar Measure

[Adamic & Adar, SN03]

- Two nodes are more similar if they share more items that are overall less frequent

$$s(a,b) = \underbrace{\sum_{i \in N(a) \cap N(b)}}_{\text{Can be any kind of shared attributes or relationships to shared entities}} \frac{1}{\log(\underbrace{\deg(i)}_{\text{Overall frequency in the data}}))}$$

Can be any kind of shared attributes or relationships to shared entities

Overall frequency in the data

Katz Score

- Two objects are similar if they are connected by shorter paths

$$s(a,b) = \sum_{l=1}^{\infty} \beta^l \cdot |\underbrace{\text{paths}^{\langle l \rangle}(a,b)}_{\text{Set of paths between a and b of length exactly } \ell}|$$

Decay factor between 0 and 1

- Since expensive to compute, often use approximate Katz, assuming some max path length of k

Duke UNIVERSITY

Personalized Page Rank

- Stationary distribution of a random walk:
  - With probability (1-c), follow a random outgoing edge
  - With probability c, jump to the target node ‘a’

SimRank [Jeh & Widom, KDD02]

- "Two objects are similar if they are related to similar objects"

- Defined as the unique solution to:

Decay factor between 0 and 1

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

Set of incoming edges into a

- Computed by iterating to convergence
- Initialization to s(a, b) = 1 if a=b and 0 otherwise

Intuition behind Simrank

- sim(a,b) measures how soon two (reverse) random walks starting from a and b meet at the same node.

- Works best for bipartite graphs (having two types of entities)

Intuition behind Simrank

Expected Distance

$$d(u,v) = \sum_{t:u \rightsquigarrow v} P[t]l(t)$$

- d(u,v) = 0, if u = v

- t: tour (path with cycles) starting at u and ending at v
- t = [w1, w2, …, wk]

$$P[t] = \prod_{i=1}^{k-1} \frac{1}{|O(w_i)|}$$

Intuition behind Simrank

Expected Meeting Distance

- expected number of steps taken for 2 random walks starting from a and b to meet.

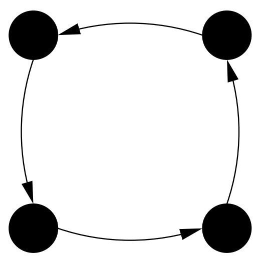
- Expected meeting distance in G is equivalent to expected distance in G^2.
  - Consider a graph $G^2 = (V \times V, E^2)$
  - There is an edge between (a,b) and (c,d) in E^2, if there are edges (a,c) and (b,d) in E

$$m(a,b) = \sum_{t:(a,b) \rightsquigarrow (x,x)} P[t]\, l(t)$$

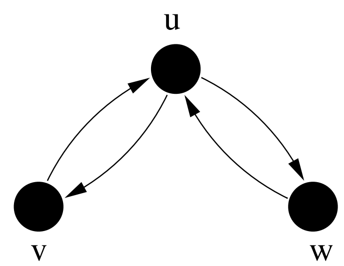
Duke UNIVERSITY

Intuition behind Simrank

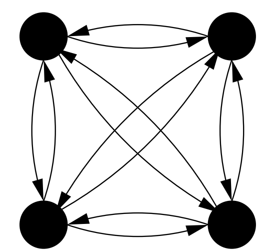
Expected Meeting Distance



m(u,v) = ∞



m(u,v) = ∞
m(u,w) = ∞
m(v,w) = 1



m(u,v) = 3

Intuition behind Simrank

Expected-f Meeting Distance

- Map distance l(t) to f(l(t), where f(z) = c^z, 0 < c < 1

$$s'(a,b) = \sum_{t:(a,b)\rightsquigarrow(x,x)} P[t]c^{l(t)}$$

- Large distances become small similarities
- Small distances become large similarities

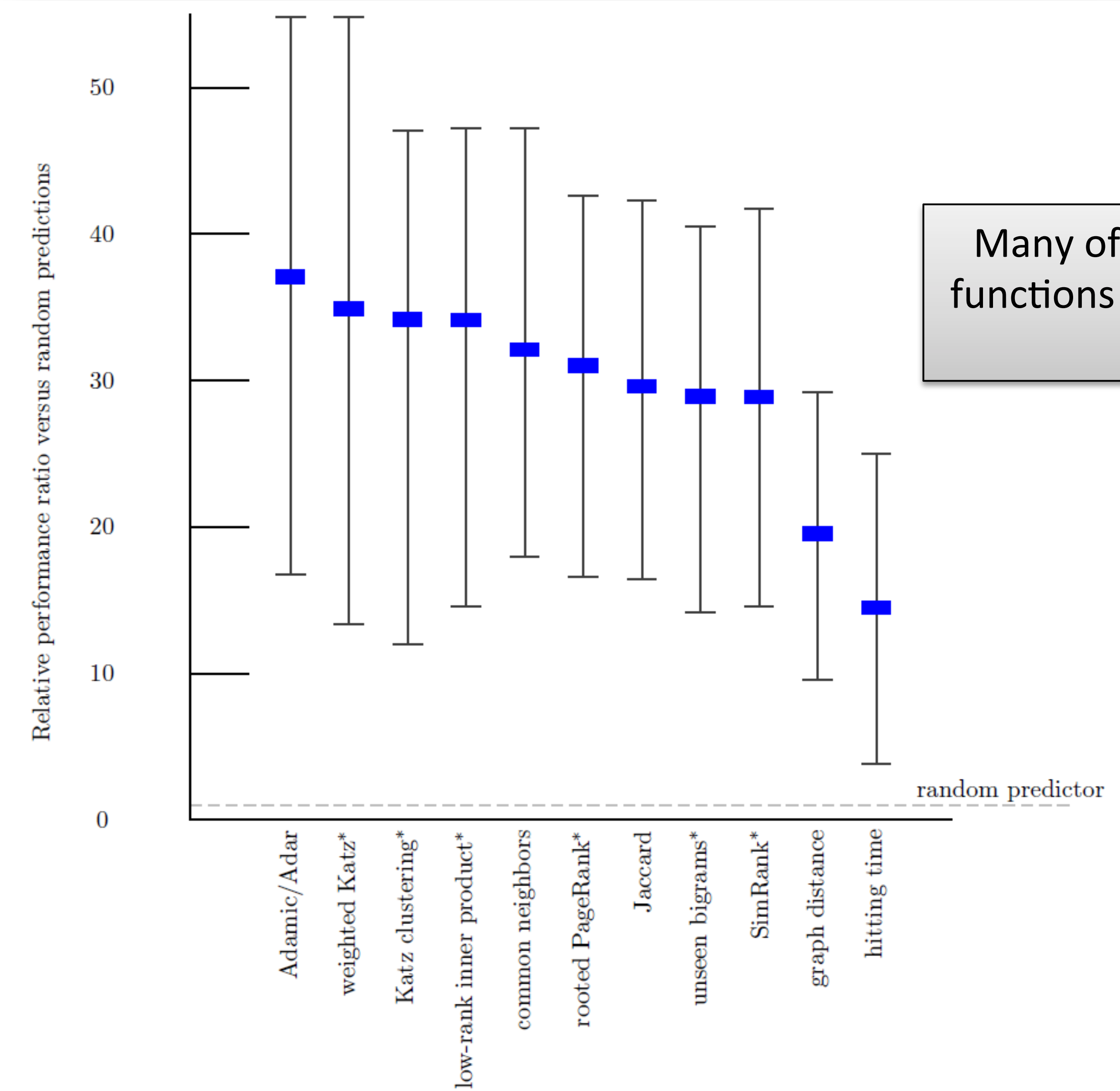
Intuition behind Simrank

- s(a,b) is equivalent to s'(a,b) where in and out edges are reversed.

$$s'(a,b) = \sum_{t:(a,b) \rightsquigarrow (x,x)} P[t]c^{l(t)}$$

$$= \sum_{c \epsilon O(a)} \sum_{d \epsilon O(b)} \sum_{t':(c,d) \rightsquigarrow (x,x)} \frac{P[t']c^{l(t')+1}}{|O(a)||O(b)|}$$

$$= \frac{c}{|O(a)||O(b)|} \sum_{c \epsilon O(a)} \sum_{d \epsilon O(b)} s'(c,d)$$



[Liben-Nowell, Kleinberg 2003]

Many of the aforementioned similarity functions are also used for link prediction in social networks

This Class

- Collective Entity Resolution for Relational Data
  - Problem Statement
  - Motivating Example
  - Similarity functions for Linked Data
  - Relational Clustering

Relational Clustering

Blocking:

- Identify similar pairs of records.

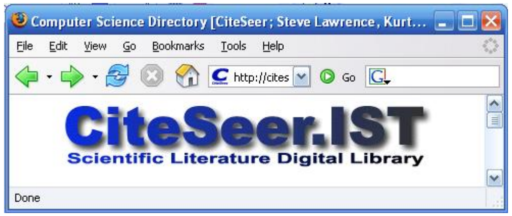
Bootstrapping:

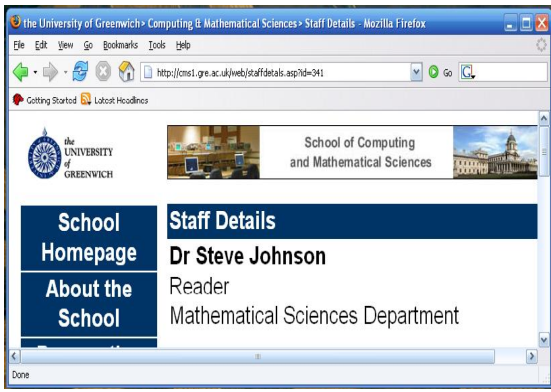
- Create some high confidence clusters of duplicate amongst blocked pairs.

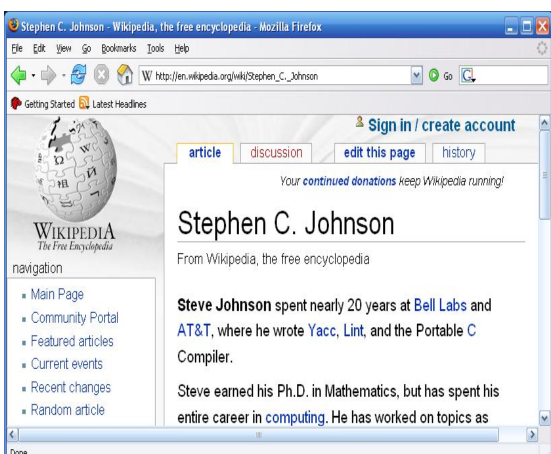
Iteration:

- Merge two closest clusters if similarity > threshold
- Update the similarities between neighboring clusters based on the fact that the cluster has been merged.

Relational Clustering using an Example







P1: "*JOSTLE: Partitioning of Unstructured Meshes for Massively Parallel Machines*", C. Walshaw, M. Cross, M. G. Everett, S. Johnson

P2: "*Partitioning Mapping of Unstructured Meshes to Parallel Machine Topologies*", C. Walshaw, M. Cross, M. G. Everett, S. Johnson, K. McManus

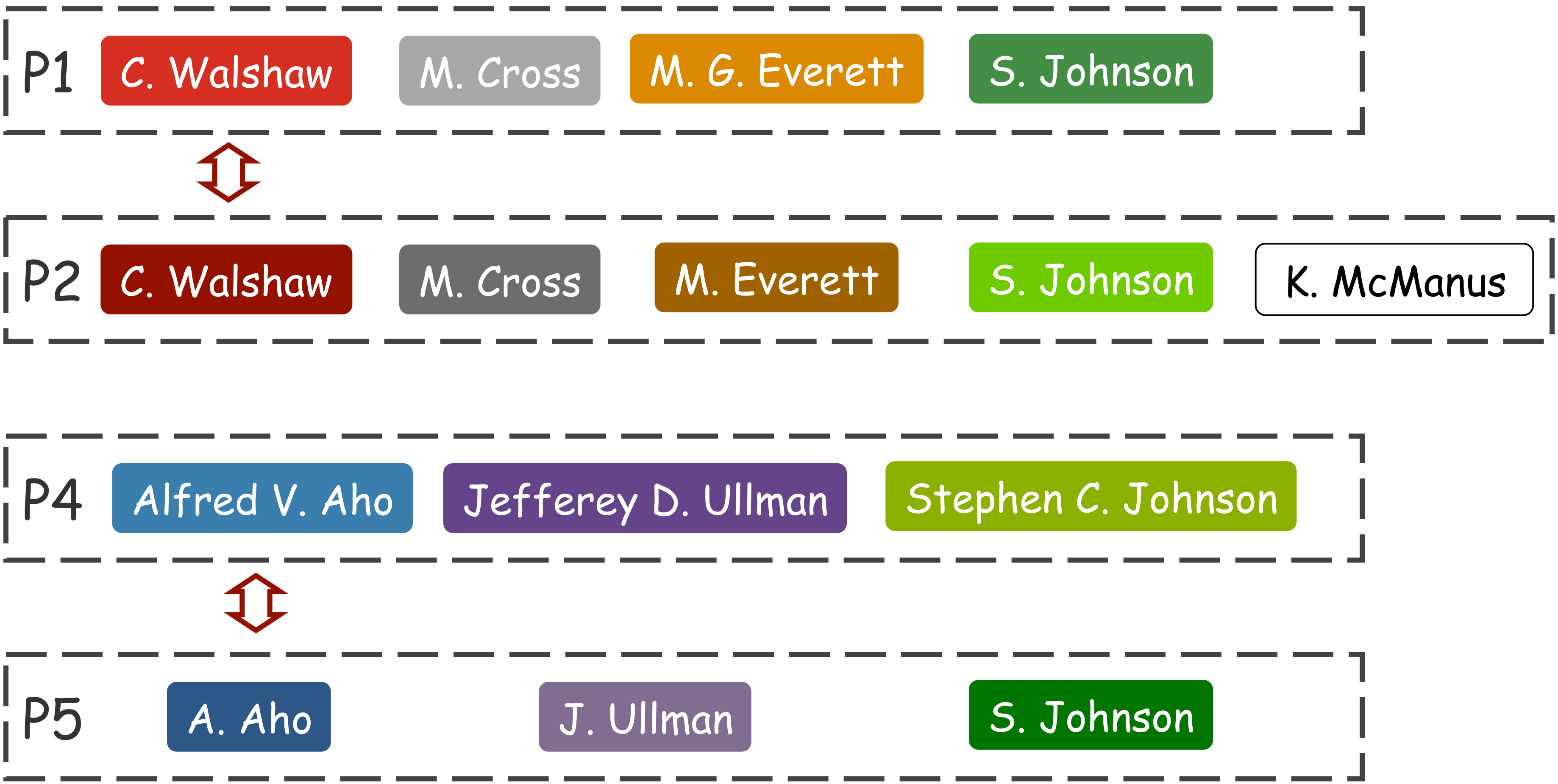
P3: "*Dynamic Mesh Partitioning: A Unied Optimisation and Load-Balancing Algorithm*", C. Walshaw, M. Cross, M. G. Everett

P4: "*Code Generation for Machines with Multiregister Operations*", Alfred V. Aho, Stephen C. Johnson, Jefferey D. Ullman

P5: "*Deterministic Parsing of Ambiguous Grammars*", A. Aho, S. Johnson, J. Ullman

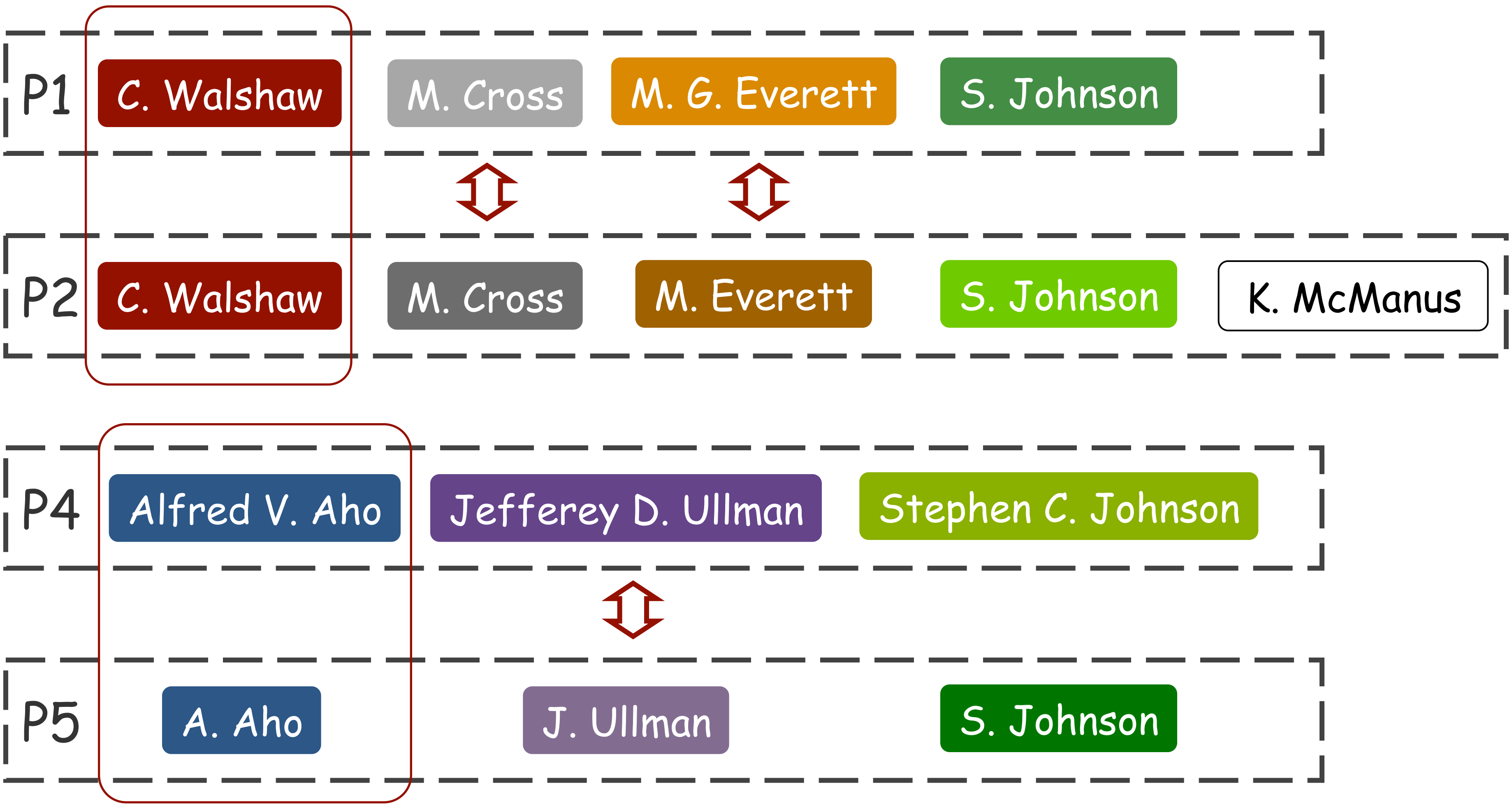
P6: "*Compilers: Principles, Techniques, and Tools*", A. Aho, R. Sethi, J. Ullman

Relational Clustering

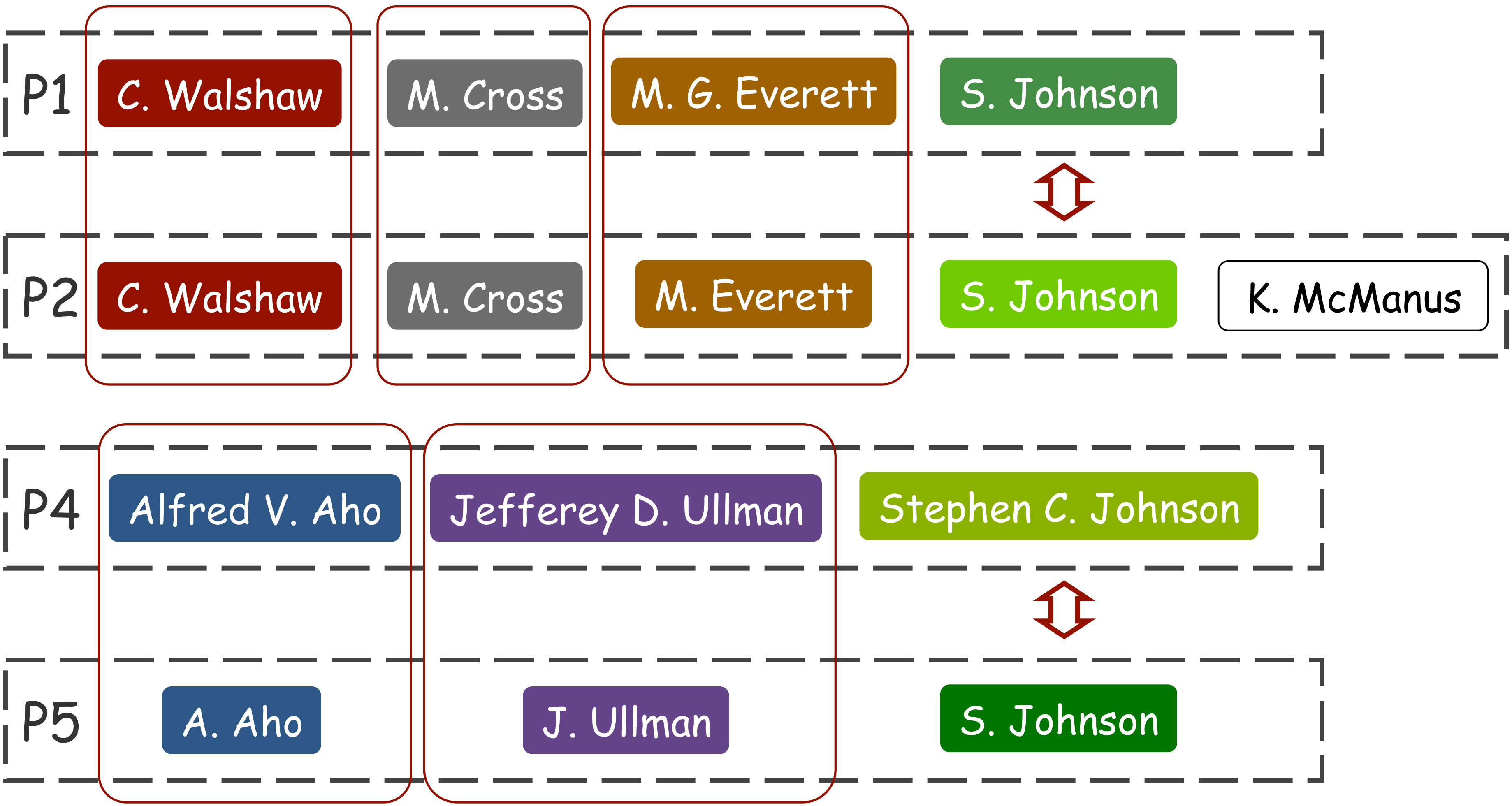


Duke UNIVERSITY

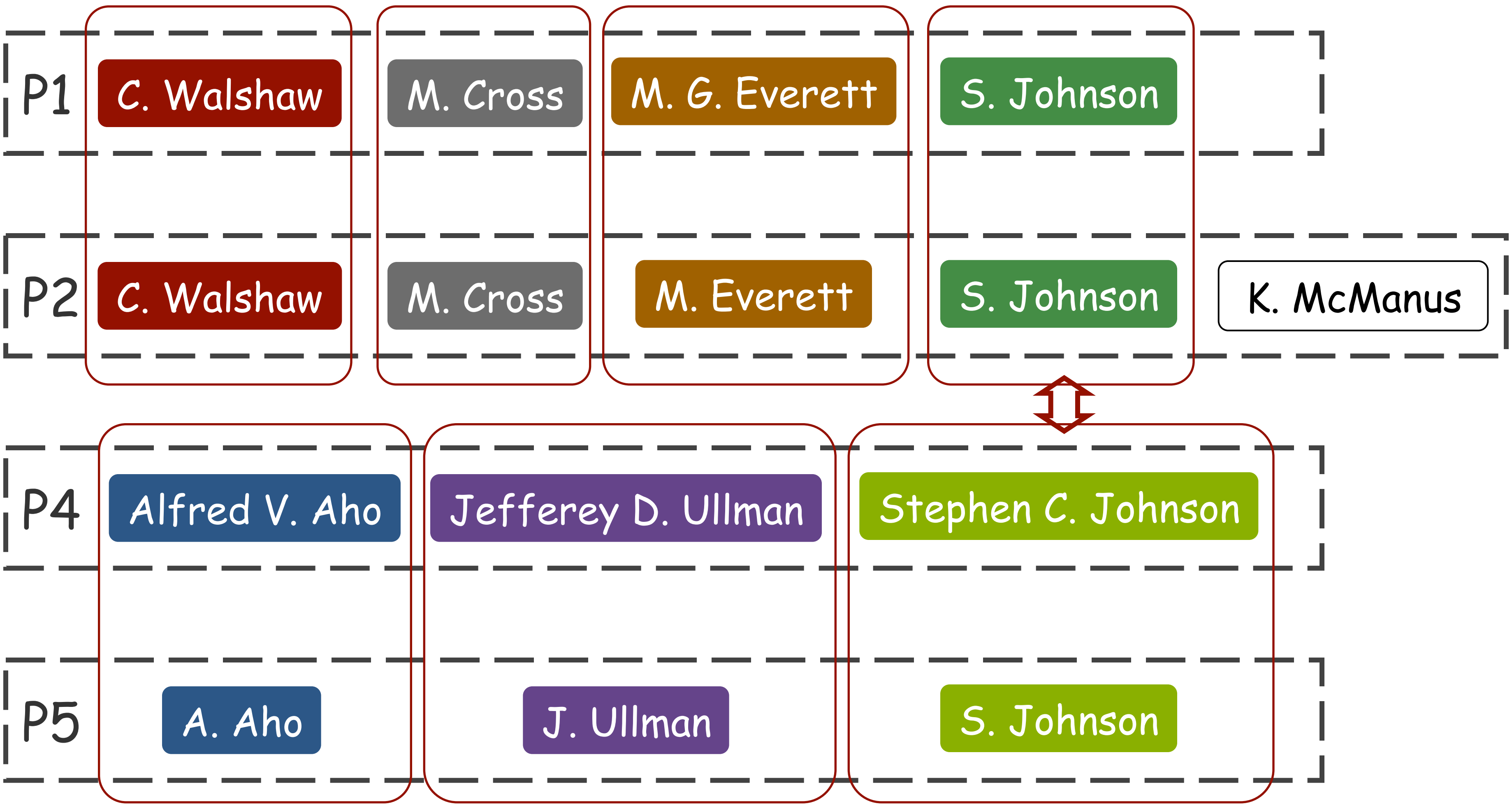
Relational Clustering



Relational Clustering



Relational Clustering



Relational Clustering

1. Find similar references using ‘blocking’
2. Bootstrap clusters using attributes and relations
3. Compute similarities for cluster pairs and insert into priority queue

4. Repeat until priority queue is empty
5. Find ‘closest’ cluster pair
6. Stop if similarity below threshold
7. Merge to create new cluster
8. Update similarity for ‘related’ clusters

- O(n k log n) algorithm w/ efficient implementation

Relational Clustering

- Never split clusters, only merge them
  - Allows efficient implementation
  - Errors early on in the process can lead to bad clustering/resolution

- Collective Resolution
  - Two objects that are not very similar can become similar if their neighbors are clustered together.

Summary

- Many similarity metrics for relational data
  - Common Neighbors
  - Adamic/Adar
  - Katz
  - Personalized Page Rank
  - Simrank

- Need collective techniques for entity resolution on linked data
  - Relational Clustering

- Next Class
  - Collective Resolution using Markov Logic
  - Scaling Collective Entity Resolution