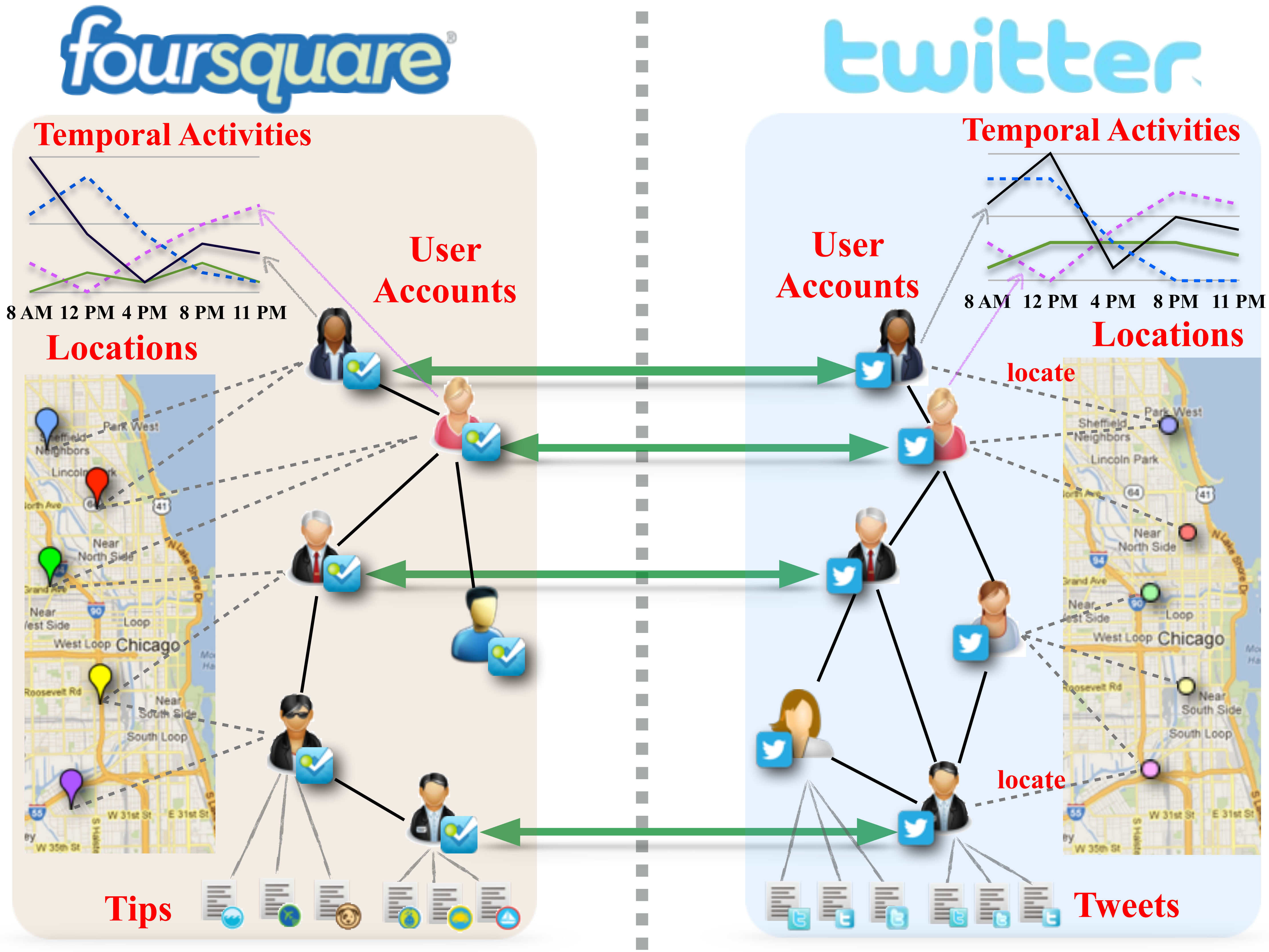
IEEE IRI 2015

PNA: Partial Network Alignment with Generic Stable Matching

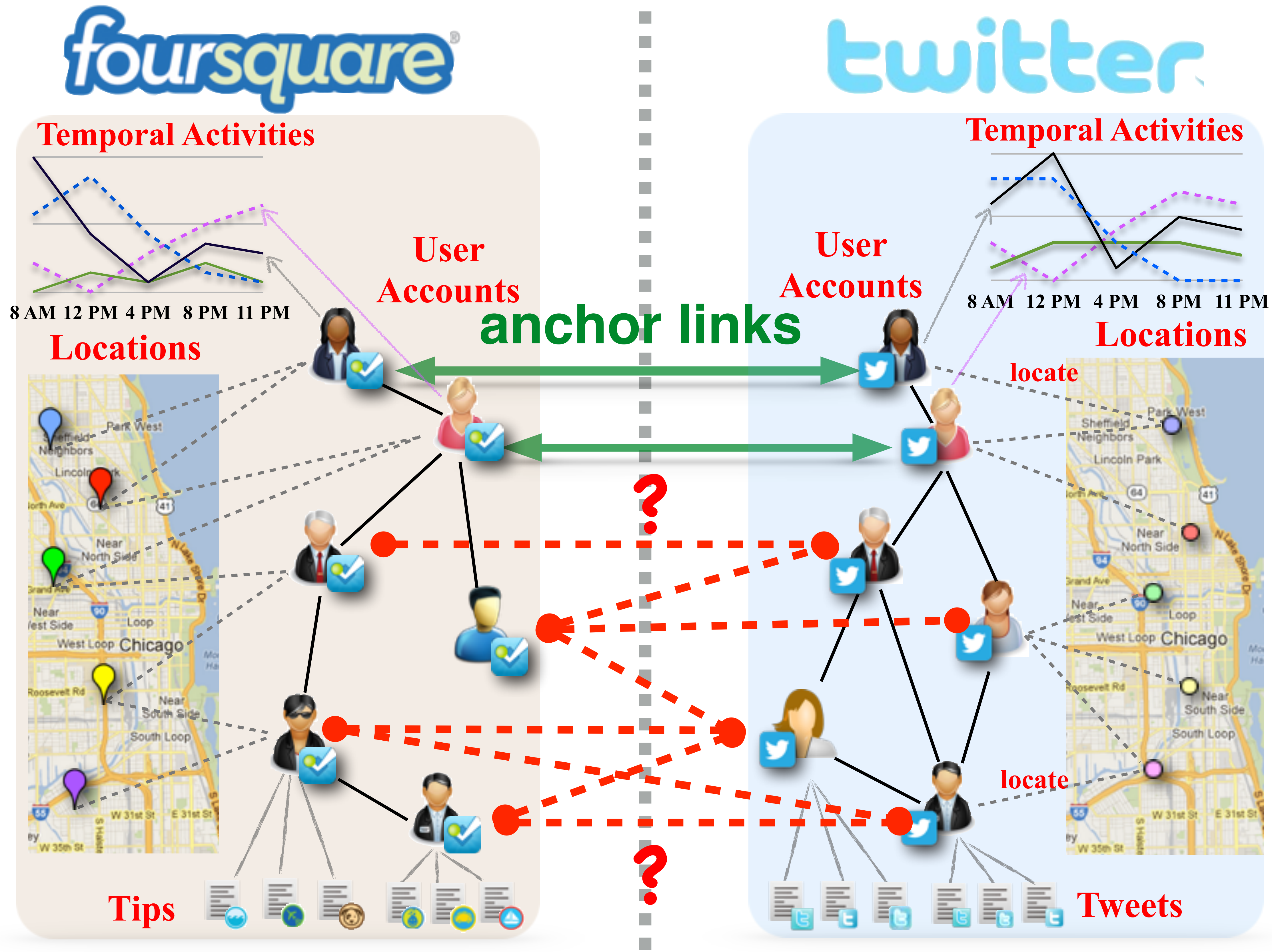
Jiawei Zhang[1], Weixiang Shao[1], Senzhang Wang[2], Xiangnan Kong[3], and Philip S. Yu[1,4]

[1] University of Illinois at Chicago, Chicago, IL, USA
[2] Beihang University, Beijing, China
[3] Worcester Polytechnic Institute, Worcester, MA, USA
[4] Institute for Data Science, Tsinghua University, Beijing, China

Users participate in multiple social networks simultaneously



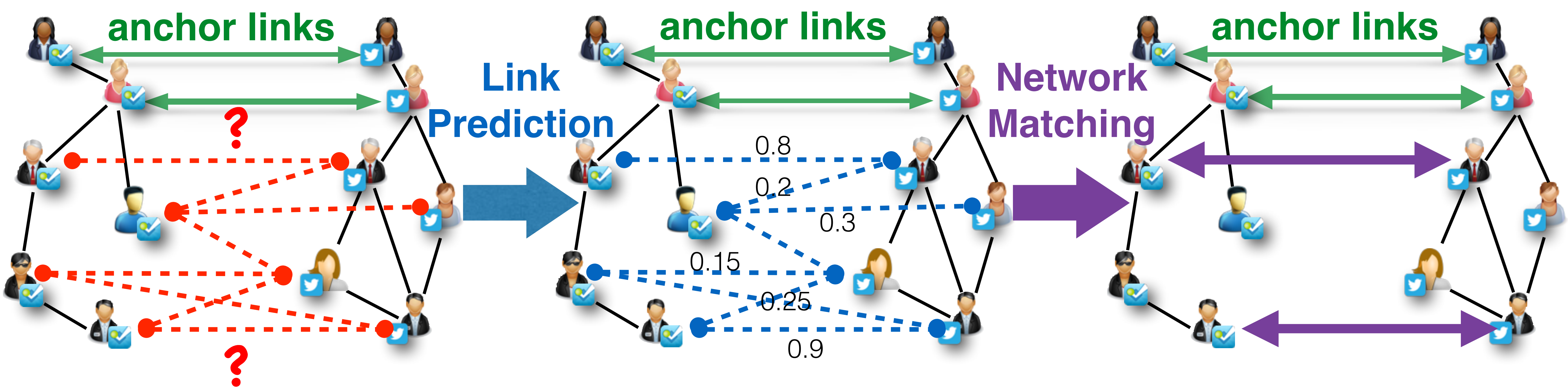
Problem Studied: Social Network Alignment via Shared Common Users



Proposed Network Alignment Framework: PNA (Partial Network Aligner)

- step 1: potential anchor link inference with information across networks

Motivations: use the heterogeneous information across social networks to infer the existence probabilities of potential anchor links.



Motivations: networks studied in this paper are partially aligned, and each user in a network can be connected to at most one user in another network.

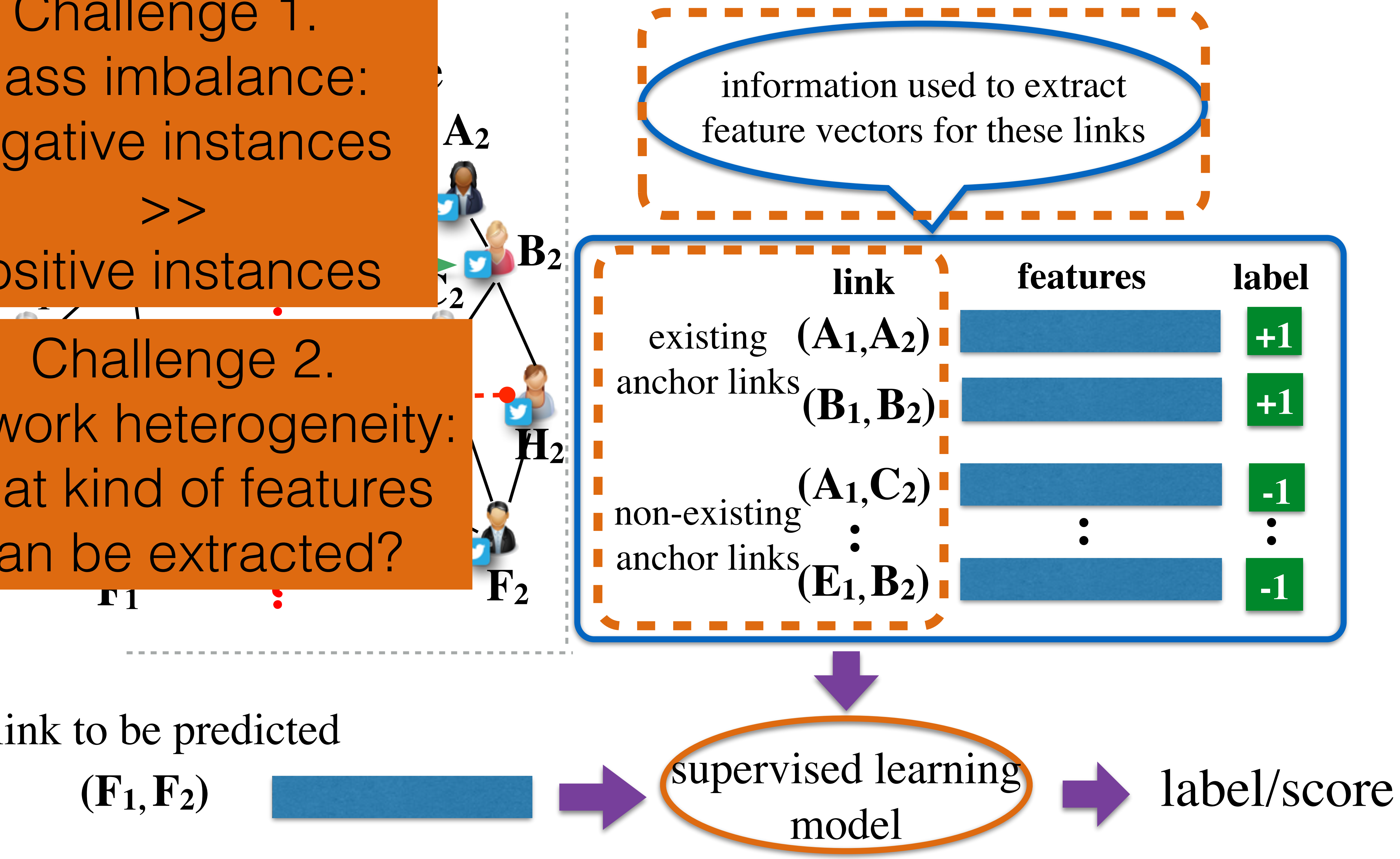
- step 2: network matching to prune redundant non-existing anchor links

Step 1: inferring potential anchor links across networks

- Proposed Method: Supervised Anchor Link Prediction

Challenge 1.
class imbalance:
negative instances
>>
positive instances

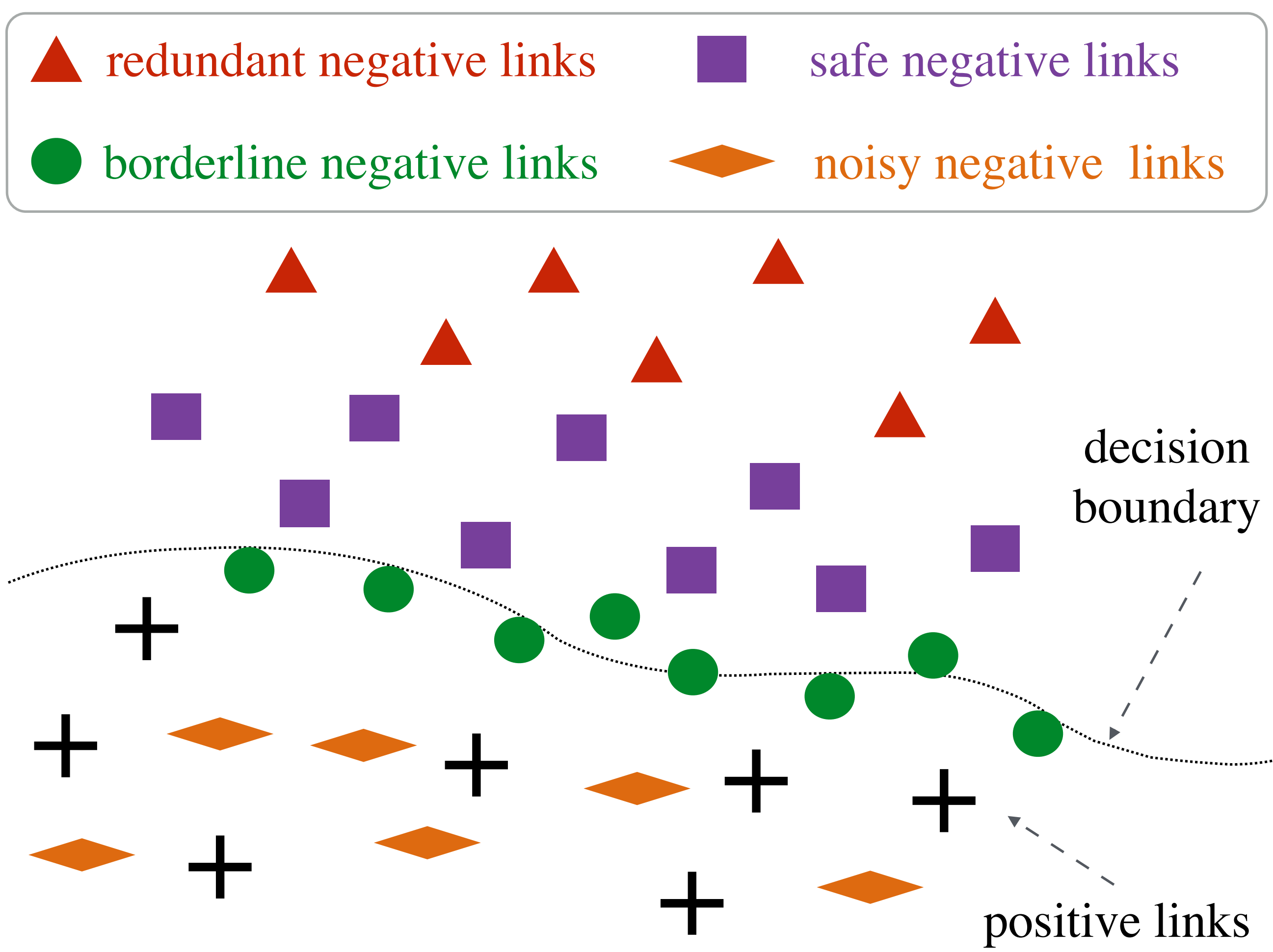
Challenge 2.
network heterogeneity:
what kind of features
can be extracted?



Challenge 1: Class Imbalance

training set | Pos | Negative

- Proposed Solution 1: **Down Sampling** the Negative Links



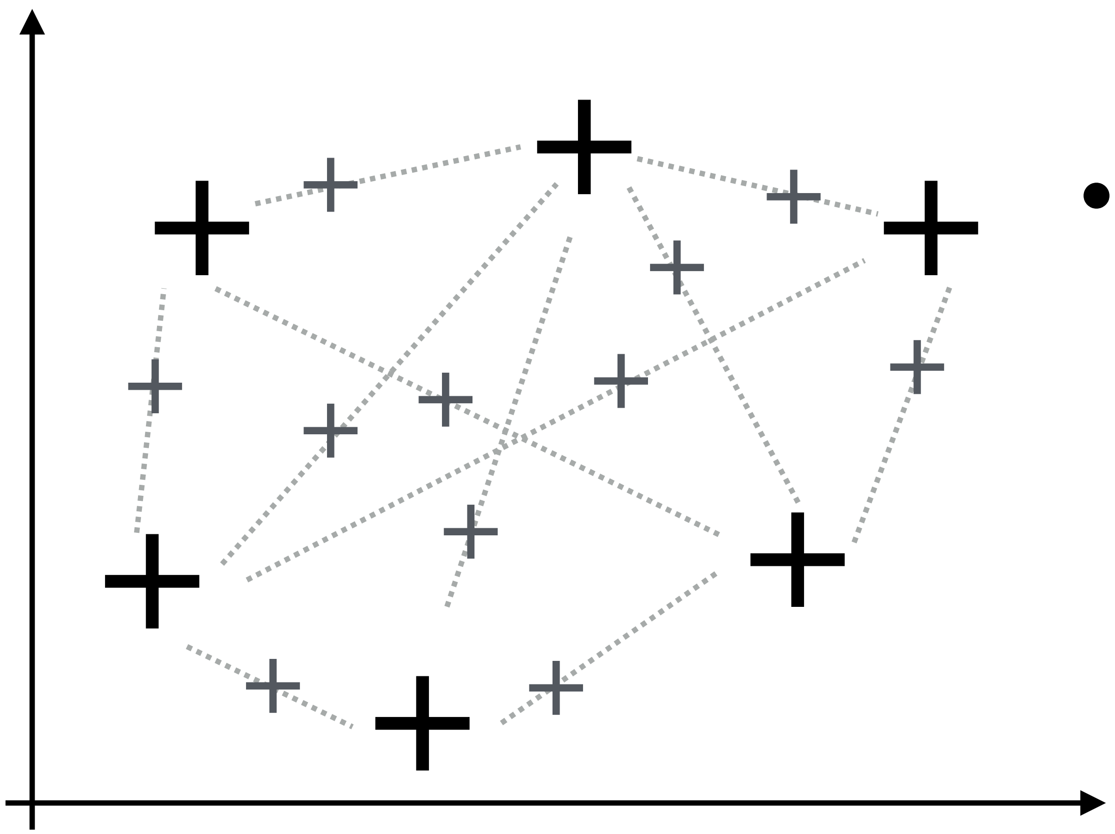
- Distributions of Negative Links in the Feature Space
  - **Safe Negative Links**
  - Borderline Negative Links
  - Noisy Negative Links
  - Redundant Negative Links

training set | Pos | Safe Neg | Borderline Negative | Noisy Negative | Redundant Negative

Challenge 1: Class Imbalance

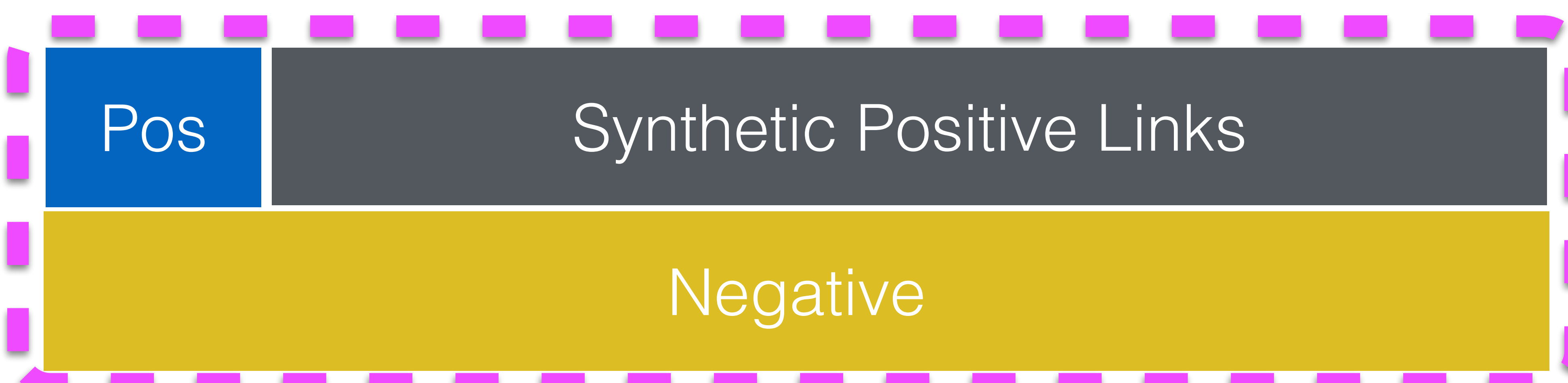
training set

- Proposed Solution 2: **Over Sampling** the Positive Links



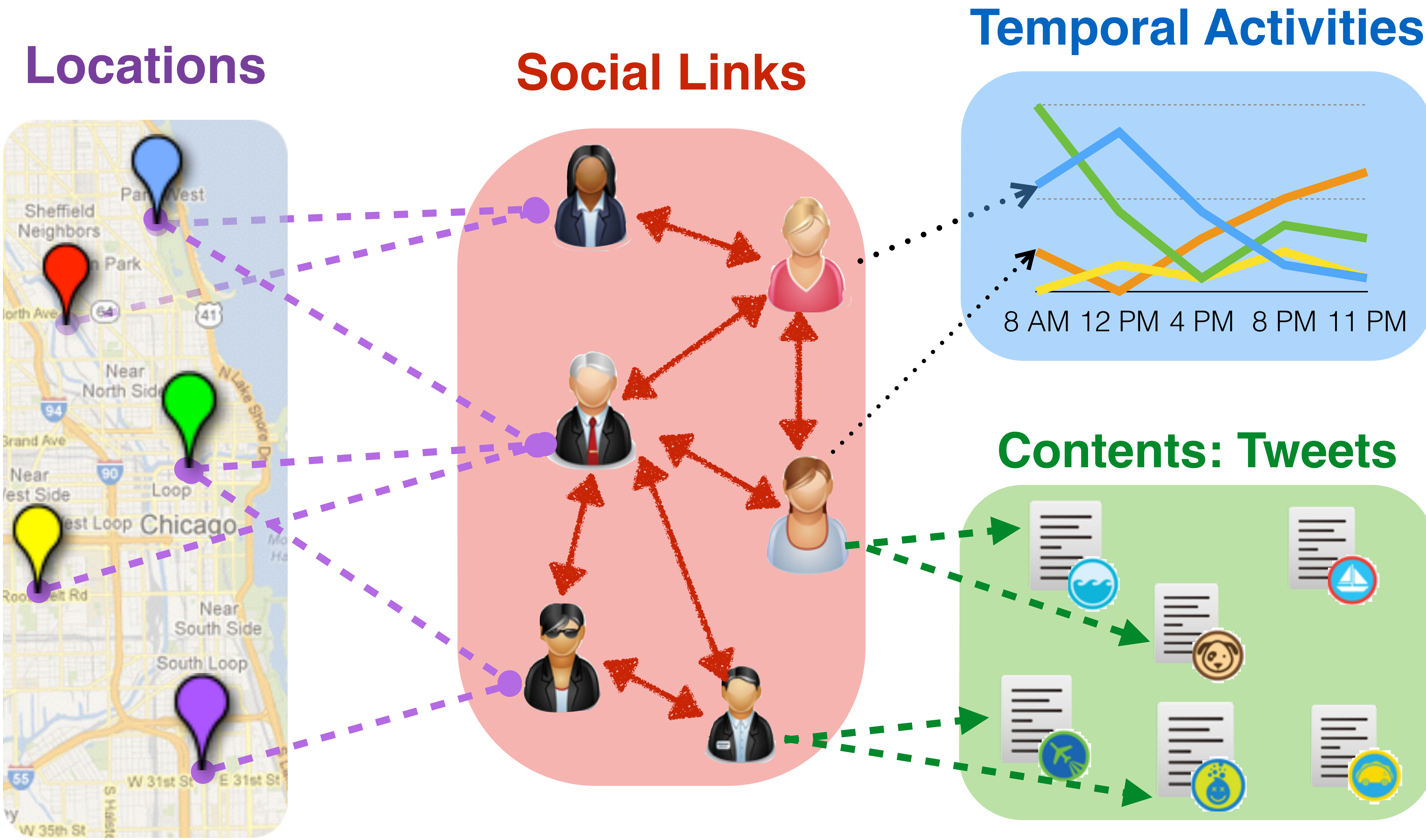
- Synthetic Positive Links Generation in the Feature Space
    - generate random synthetic positive instances between pairs of positive instances in the feature space

training set



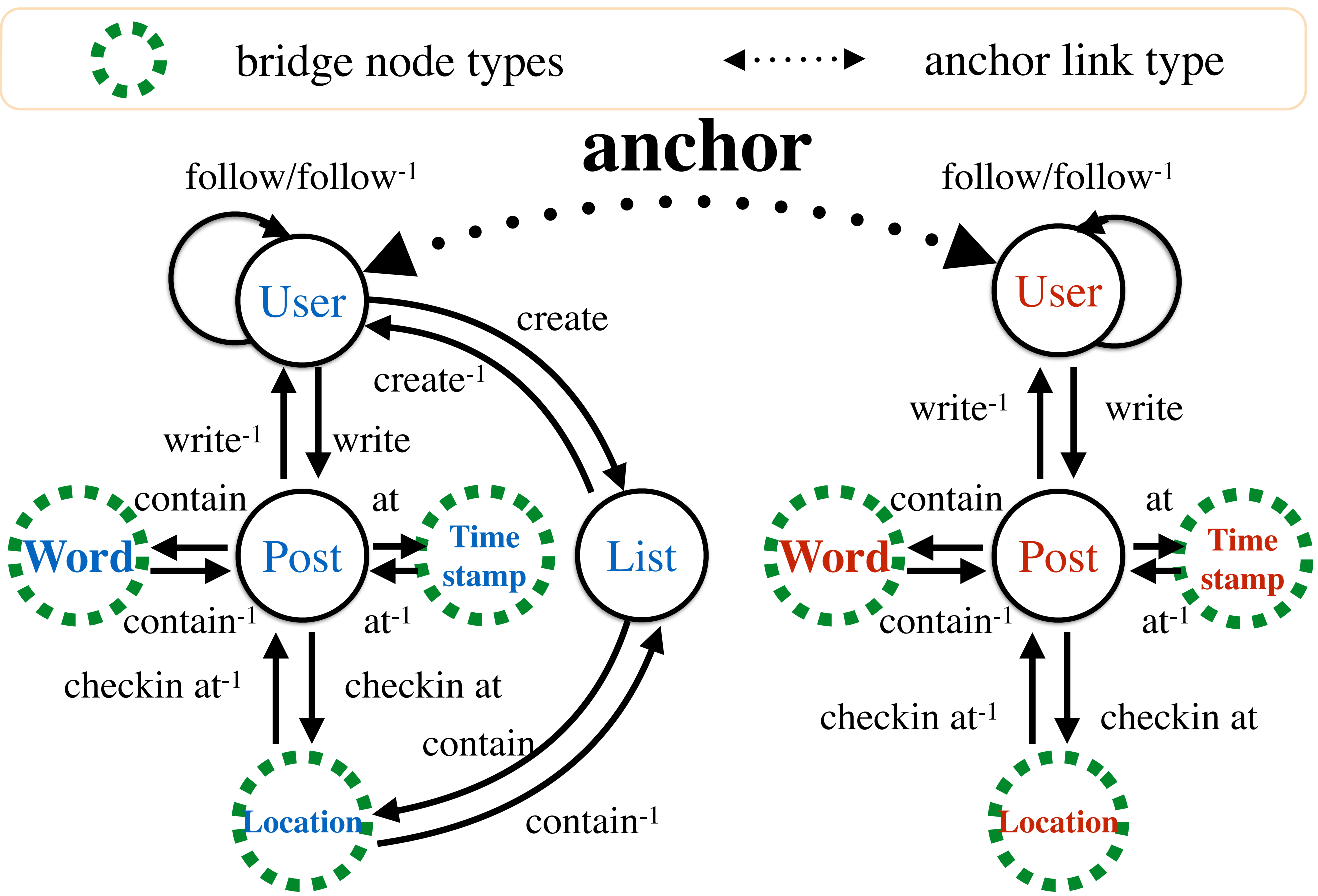
Challenge 2: Network Heterogeneity & Feature Extraction

Information Types: **Who** **Where** **What** **When**



Proposed Solution: Anchor Meta Paths

bridge node: nodes (besides users) shared across networks



Schema of Network$^{(i)}$ Schema of Network$^{(j)}$

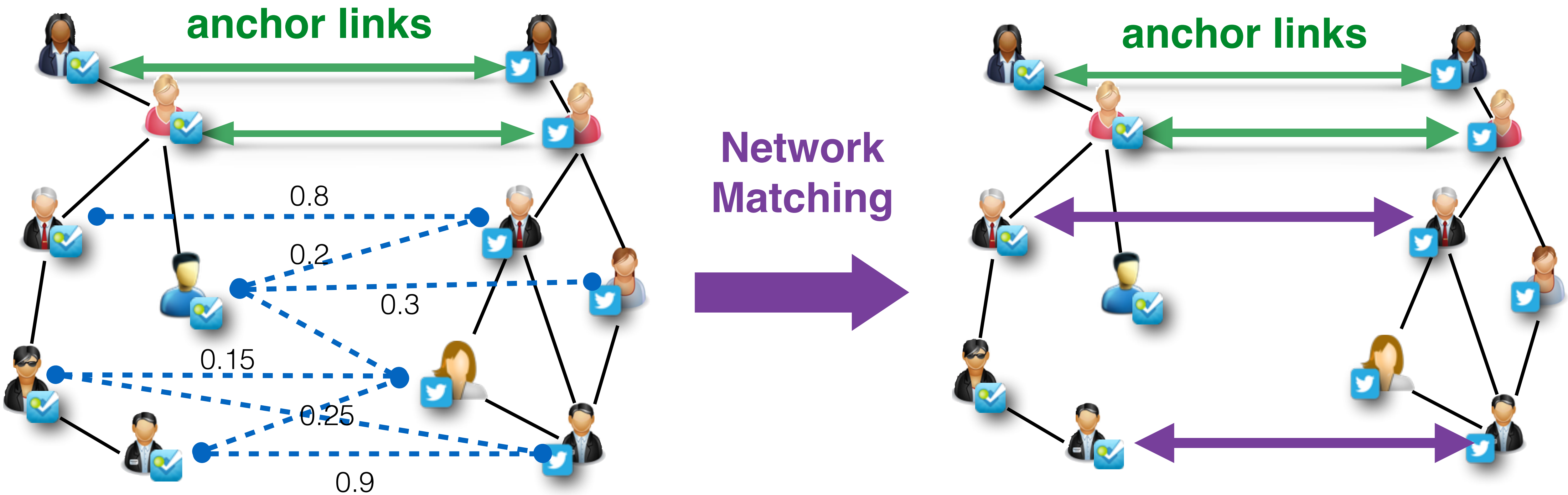
- feature extracted for anchor link ($u^{(i)}$, $v^{(j)}$) based on anchor meta path Ψ
  - number of anchor meta path instances connecting $u^{(i)}$ and $v^{(j)}$

- *Common Out Neighbor Anchor Meta Path* (Ψ_1): $User^{(i)} \xrightarrow{follow} User^{(i)} \xleftrightarrow{Anchor} User^{(j)} \xleftarrow{follow} User^{(j)}$ or "$\mathcal{U}^{(i)} \rightarrow \mathcal{U}^{(i)} \leftrightarrow \mathcal{U}^{(j)} \leftarrow \mathcal{U}^{(j)}$" for short.
- *Common In Neighbor Anchor Meta Path* (Ψ_2): $User^{(i)} \xleftarrow{follow} User^{(i)} \xleftrightarrow{Anchor} User^{(j)} \xrightarrow{follow} User^{(j)}$ or "$\mathcal{U}^{(i)} \leftarrow \mathcal{U}^{(i)} \leftrightarrow \mathcal{U}^{(j)} \rightarrow \mathcal{U}^{(j)}$" .
- *Common Out In Neighbor Anchor Meta Path* (Ψ_3): $User^{(i)} \xrightarrow{follow} User^{(i)} \xleftrightarrow{Anchor} User^{(j)} \xrightarrow{follow} User^{(j)}$ or "$\mathcal{U}^{(i)} \rightarrow \mathcal{U}^{(i)} \leftrightarrow \mathcal{U}^{(j)} \rightarrow \mathcal{U}^{(j)}$".
- *Common In Out Neighbor Anchor Meta Path* (Ψ_4): $User^{(i)} \xleftarrow{follow} User^{(i)} \xleftrightarrow{Anchor} User^{(j)} \xleftarrow{follow} User^{(j)}$ or "$\mathcal{U}^{(i)} \leftarrow \mathcal{U}^{(i)} \leftrightarrow \mathcal{U}^{(j)} \leftarrow \mathcal{U}^{(j)}$".

- *Common Location Checkin Anchor Meta Path 1* (Ψ_5): $User^{(i)} \xrightarrow{write} Post^{(i)} \xrightarrow{checkin\ at} Location \xleftarrow{checkin\ at} Post^{(j)} \xleftarrow{write} User^{(j)}$ or "$\mathcal{U}^{(i)} \rightarrow \mathcal{P}^{(i)} \rightarrow \mathcal{L} \leftarrow \mathcal{P}^{(j)} \leftarrow \mathcal{U}^{(j)}$".
- *Common Location Checkin Anchor Meta Path 2* (Ψ_6): $User^{(i)} \xrightarrow{create} List^{(i)} \xrightarrow{contain} Location \xleftarrow{checkin\ at} Post^{(j)} \xleftarrow{write} User^{(j)}$ or "$\mathcal{U}^{(i)} \rightarrow \mathcal{I}^{(i)} \rightarrow \mathcal{L} \leftarrow \mathcal{P}^{(j)} \leftarrow \mathcal{U}^{(j)}$".
- *Common Timestamps Anchor Meta Path* (Ψ_7): $User^{(i)} \xrightarrow{write} Post^{(i)} \xrightarrow{at} Time \xleftarrow{at} Post^{(j)} \xleftarrow{write} User^{(j)}$ or "$\mathcal{U}^{(i)} \rightarrow \mathcal{P}^{(i)} \rightarrow \mathcal{T} \leftarrow \mathcal{P}^{(j)} \leftarrow \mathcal{U}^{(j)}$".
- *Common Word Usage Anchor Meta Path* (Ψ_8): $User^{(i)} \xrightarrow{write} Post^{(i)} \xrightarrow{contain} Word \xleftarrow{contain} Post^{(j)} \xleftarrow{write} User^{(j)}$ or "$\mathcal{U}^{(i)} \rightarrow \mathcal{P}^{(i)} \rightarrow \mathcal{W} \leftarrow \mathcal{P}^{(j)} \leftarrow \mathcal{U}^{(j)}$".

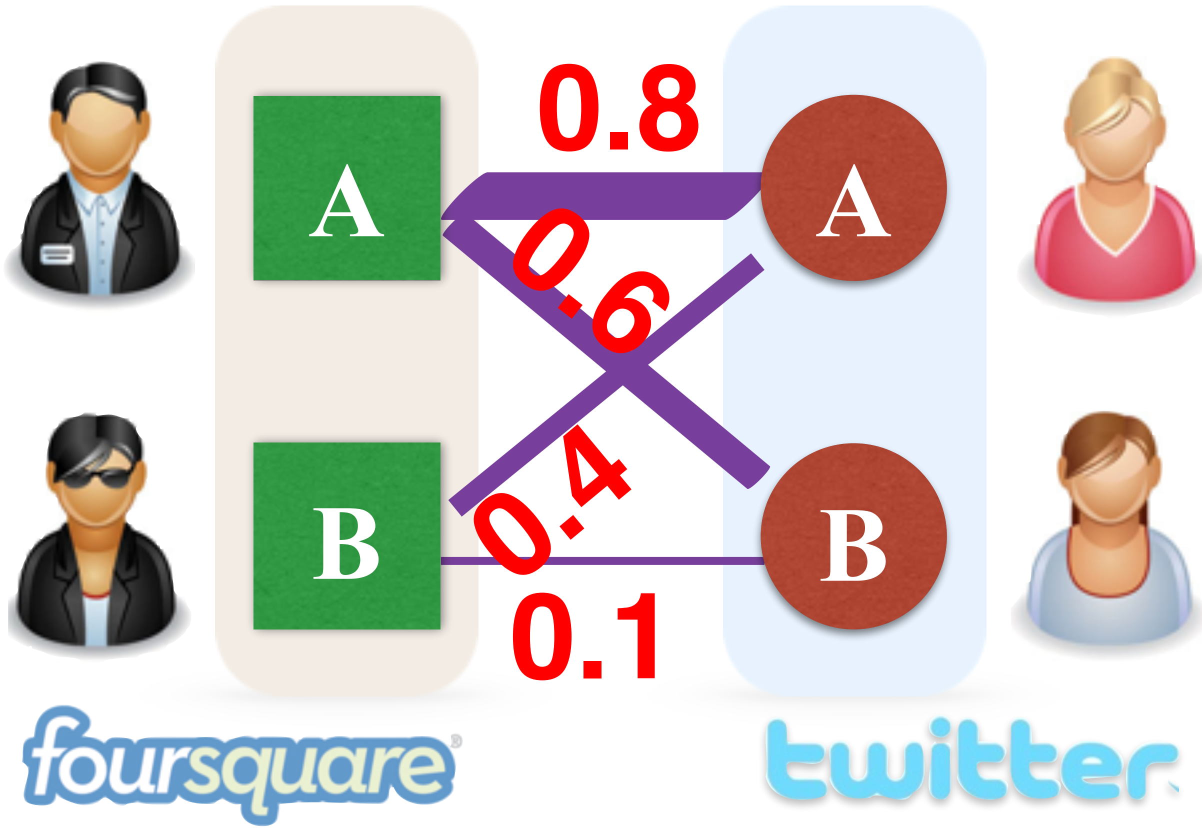
$$\text{score}_{\Psi}(u^{(i)}, v^{(j)}) = \left| \{\psi | (\psi \in \Psi) \wedge (u^{(i)} \in T_1) \wedge (v^{(j)} \in T_k)\} \right|$$

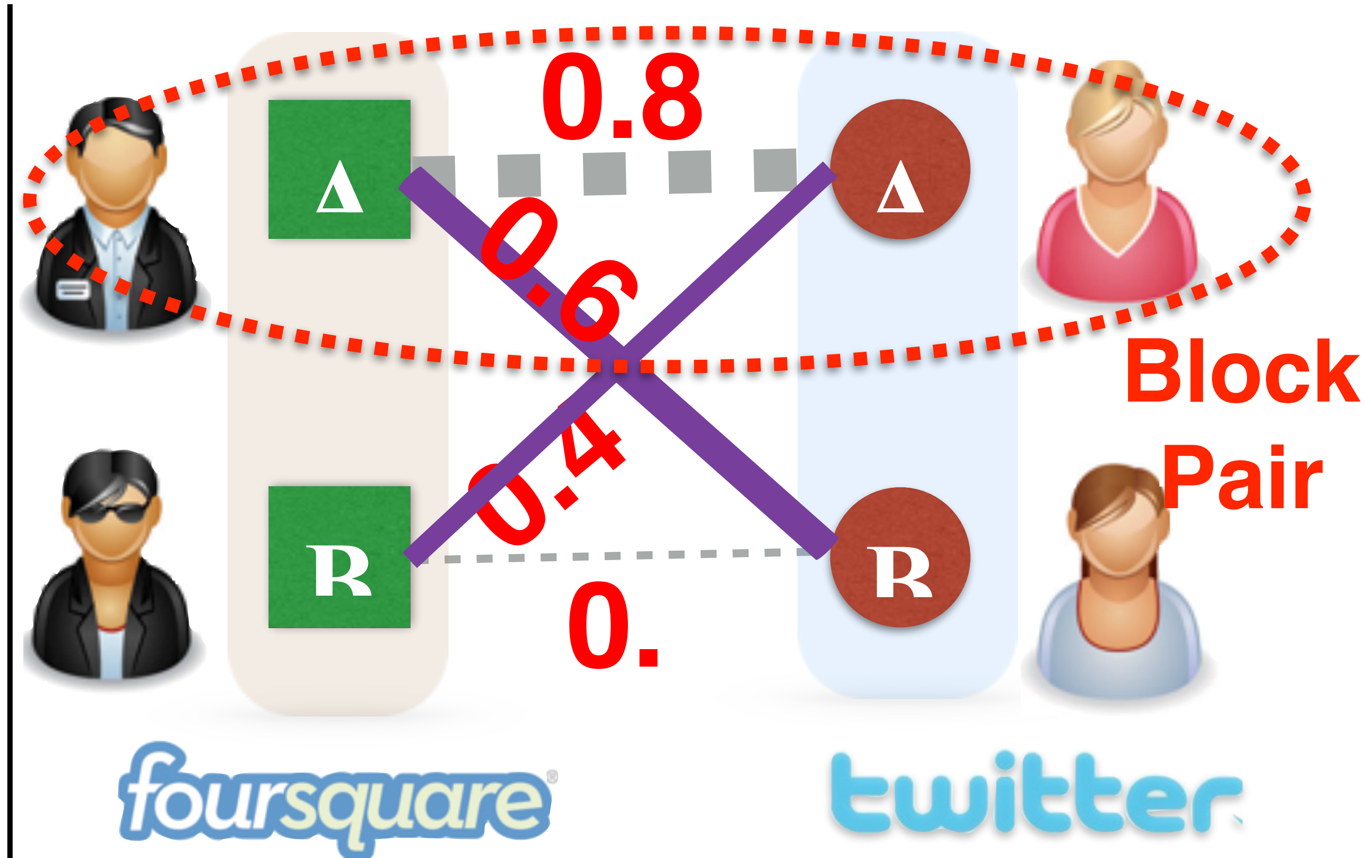
Step 2: Network Matching to Prune Non-existing Anchor Links

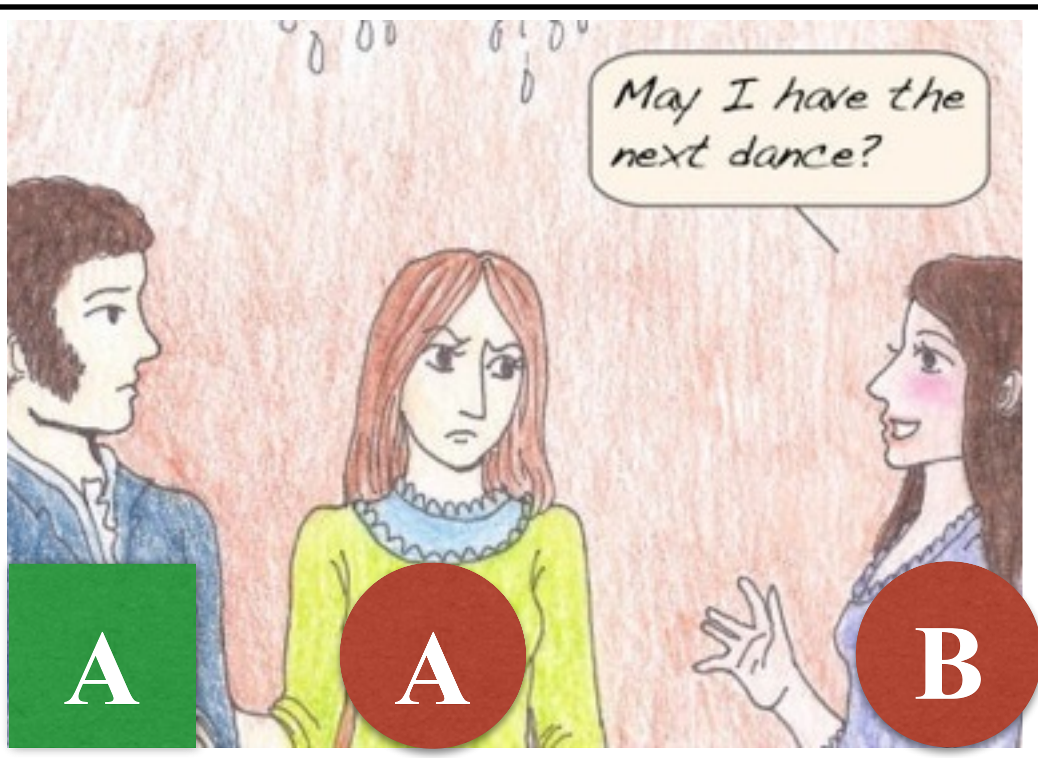


- Motivations:
  - constraint on anchor links is **1-to-1**, according to existing works
  - networks studied in this paper are partially aligned, many users are not connected to anchor links
  - revised constraint on anchor links is **1-to-1≤** (one-to-at most one)
  - how to keep the 1-to-1≤ constraint and prune redundant non-existing anchor links is very challenging

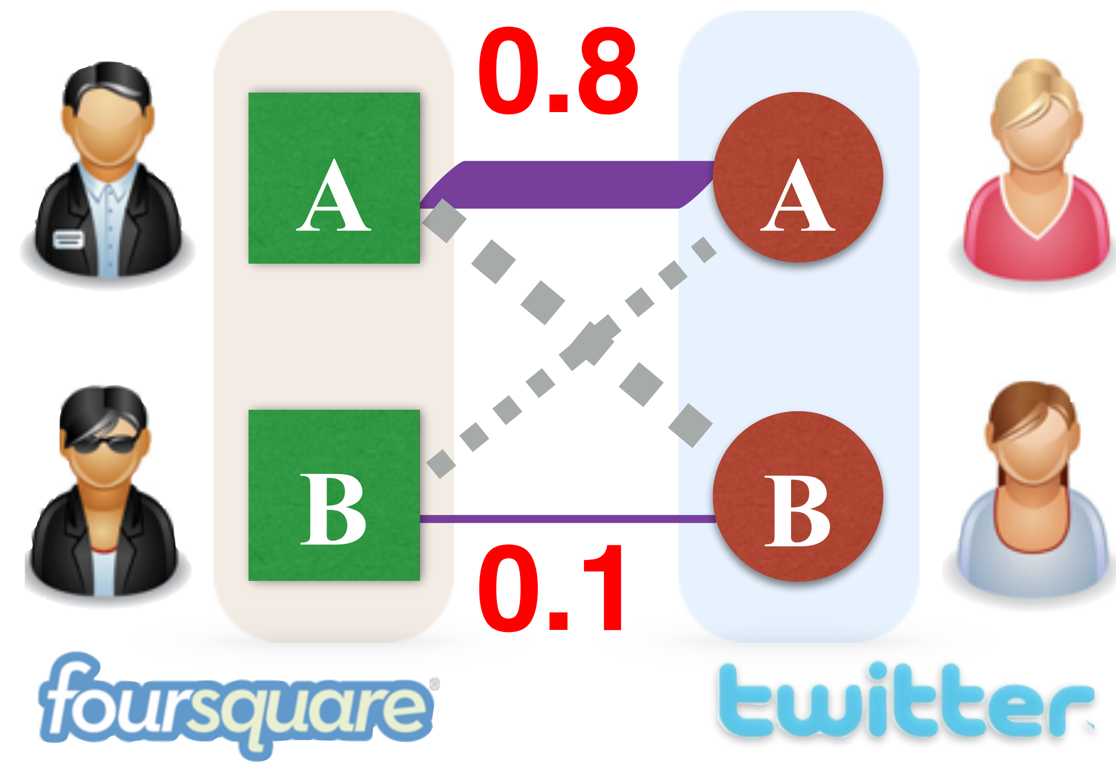
Proposed Solution of **1-to-1** Constraint: **Stable Matching**





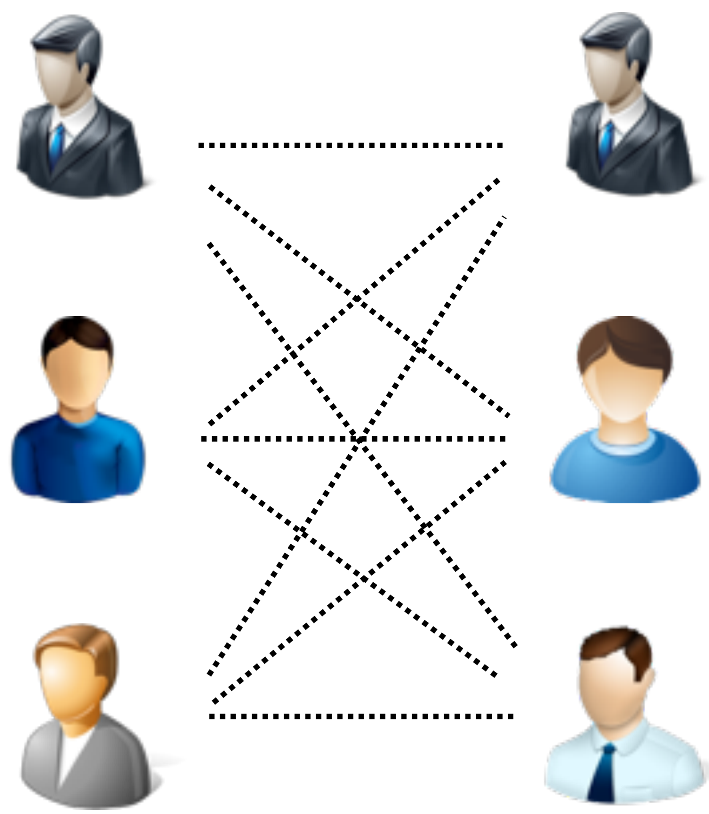


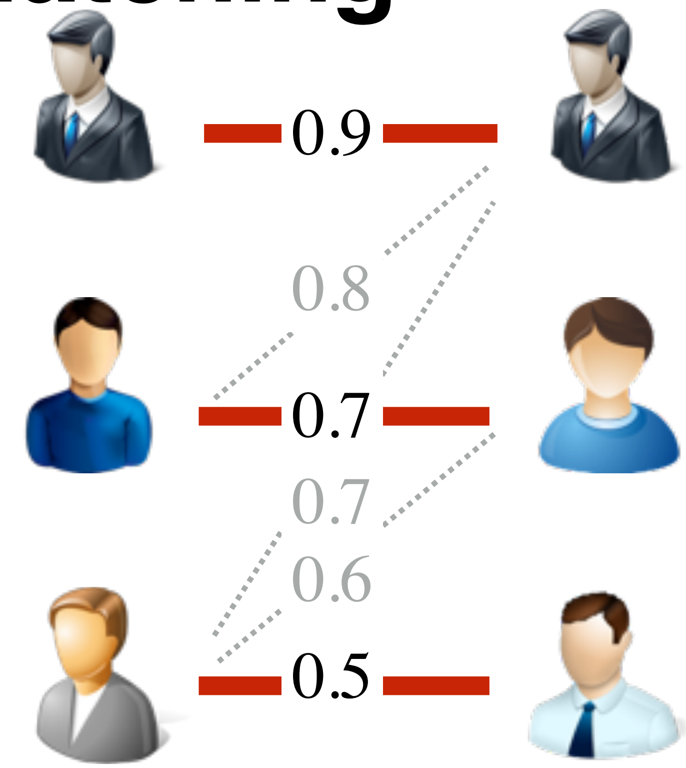
Matching with Block Pair is unstable



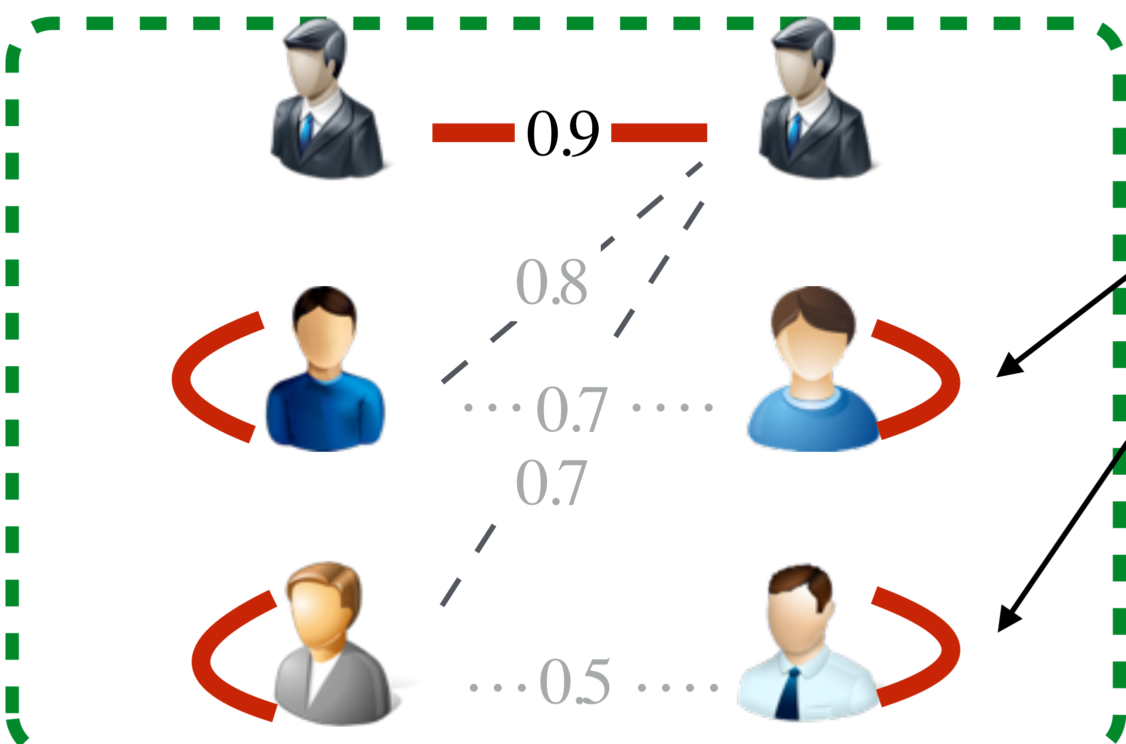
Stable Matching

Proposed Solution of **1-to-1≤** Constraint: **Self Matching** and **Generic Stable Matching**





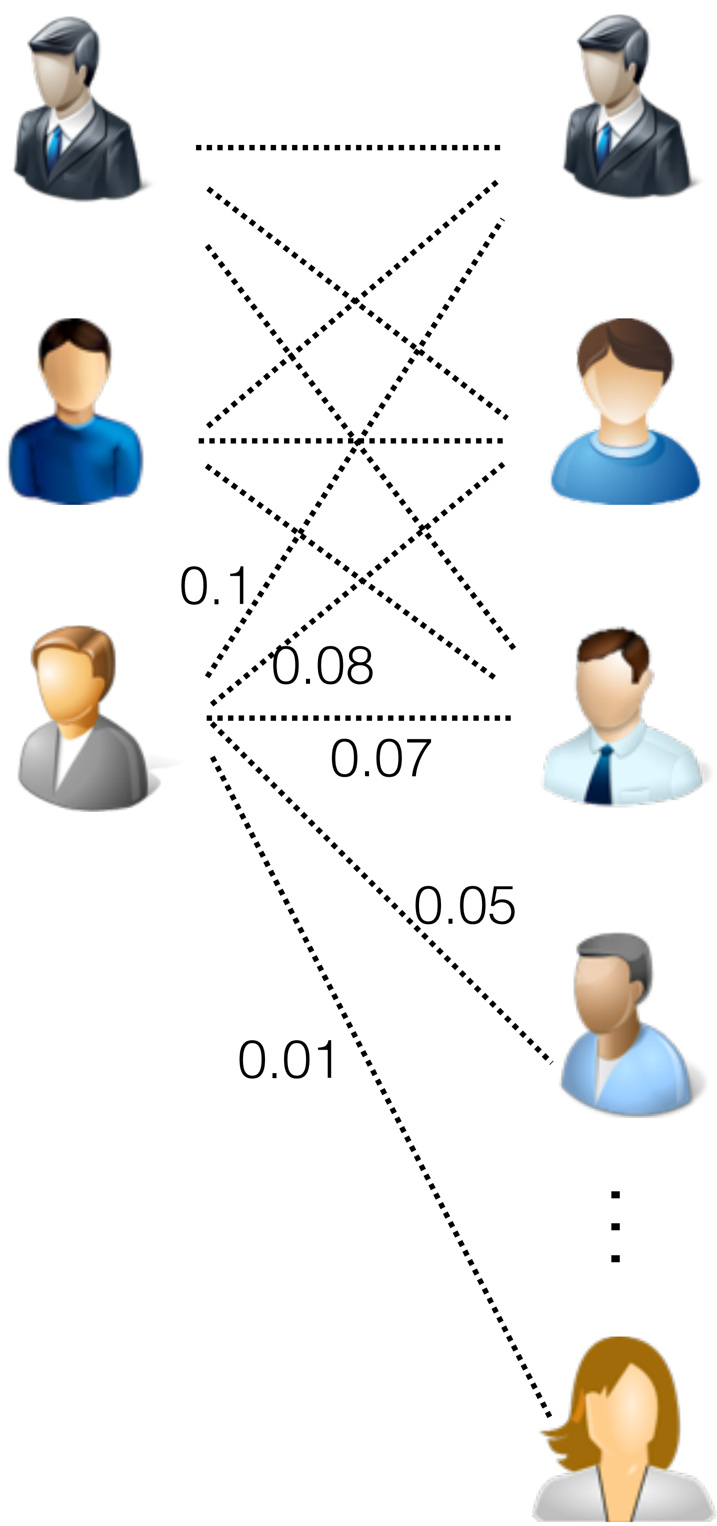
stable matching result

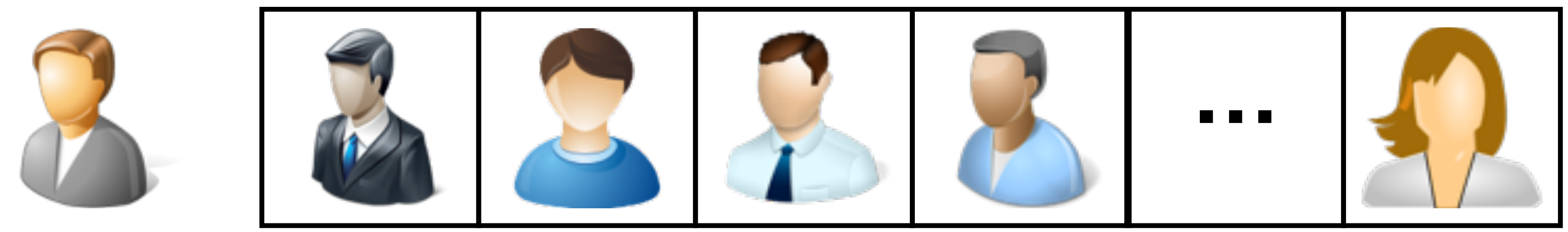


Generic stable matching result

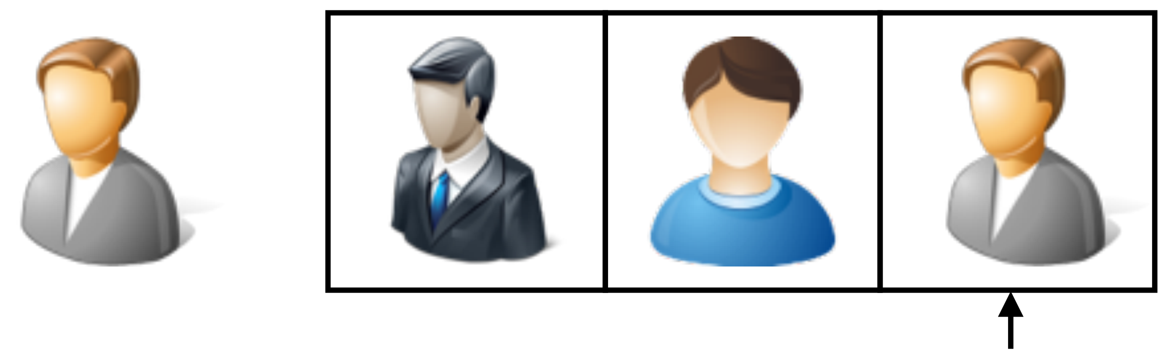
- *Self Matching*: users who are shared common users prefer to stay unconnected
- *Generic Stable Matching*: Stable matching (for shared users) which also allows self matching (unshared users)
- How to do self matching and generic stable matching?

Self Matching and Generic Stable Matching





Preference List



place the user himself at the $(K+1)_{th}$ entry

Truncated Preference List

- K: partial matching rate, used to control the length of users' preference list, whose sensitivity analysis will be given in the experiments

Pseudocode of Generic Stable Matching of Networks

Algorithm 1 Generic Gale-Shapley Algorithm

Input: user sets of aligned networks: $\mathcal{U}^{(1)}$ and $\mathcal{U}^{(2)}$.
classification results of potential anchor links in $\mathcal{L}$
known anchor links in $\mathcal{A}^{(1,2)}$
truncation rate K

Output: a set of inferred anchor links $\mathcal{L}'$

Initialize the preference lists of users in $\mathcal{U}^{(1)}$ and $\mathcal{U}^{(2)}$ with predicted existence probabilities of links in $\mathcal{L}$ and known anchor links in $\mathcal{A}^{(1,2)}$, whose existence probabilities are 1.0
construct the truncated strategies from the preference lists
Initialize all users in $\mathcal{U}^{(1)}$ and $\mathcal{U}^{(2)}$ as *free*
$\mathcal{L}' = \emptyset$
while $\exists$ *free* $u_i^{(1)}$ in $\mathcal{U}^{(1)}$ and $u_i^{(1)}$'s truncated strategy is non-empty **do**
  Remove the top-ranked account $u_j^{(2)}$ from $u_i^{(1)}$'s truncated strategy
  if $u_j^{(2)} == u_i^{(1)}$ **then**
    $\mathcal{L}' = \mathcal{L}' \cup \{(u_i^{(1)}, u_i^{(1)})\}$
    Set $u_i^{(1)}$ as *stay unconnected*
  else
    if $u_j^{(2)}$ is *free* **then**
      $\mathcal{L}' = \mathcal{L}' \cup \{(u_i^{(1)}, u_j^{(2)})\}$
      Set $u_i^{(1)}$ and $u_j^{(2)}$ as *occupied*
    else
      $\exists u_p^{(1)}$ that $u_j^{(2)}$ is occupied with.
      if $u_j^{(2)}$ prefers $u_i^{(1)}$ to $u_p^{(1)}$ **then**
        $\mathcal{L}' = (\mathcal{L}' - \{(u_p^{(1)}, u_j^{(2)})\}) \cup \{(u_i^{(1)}, u_j^{(2)})\}$
        Set $u_p^{(1)}$ as *free* and $u_i^{(1)}$ as *occupied*
      end if
    end if
  end if
end while

Dataset

TABLE I
PROPERTIES OF THE HETEROGENEOUS NETWORKS

		network	
	property	**Twitter**	**Foursquare**
# node	user	5,223	5,392
	tweet/tip	9,490,707	48,756
	location	297,182	38,921
# link	friend/follow	164,920	76,972
	write	9,490,707	48,756
	locate	615,515	48,756

- # anchor links: 3,388
- Ground truth: existing anchor links
    - Hide part of the anchor links, and build models to discover them

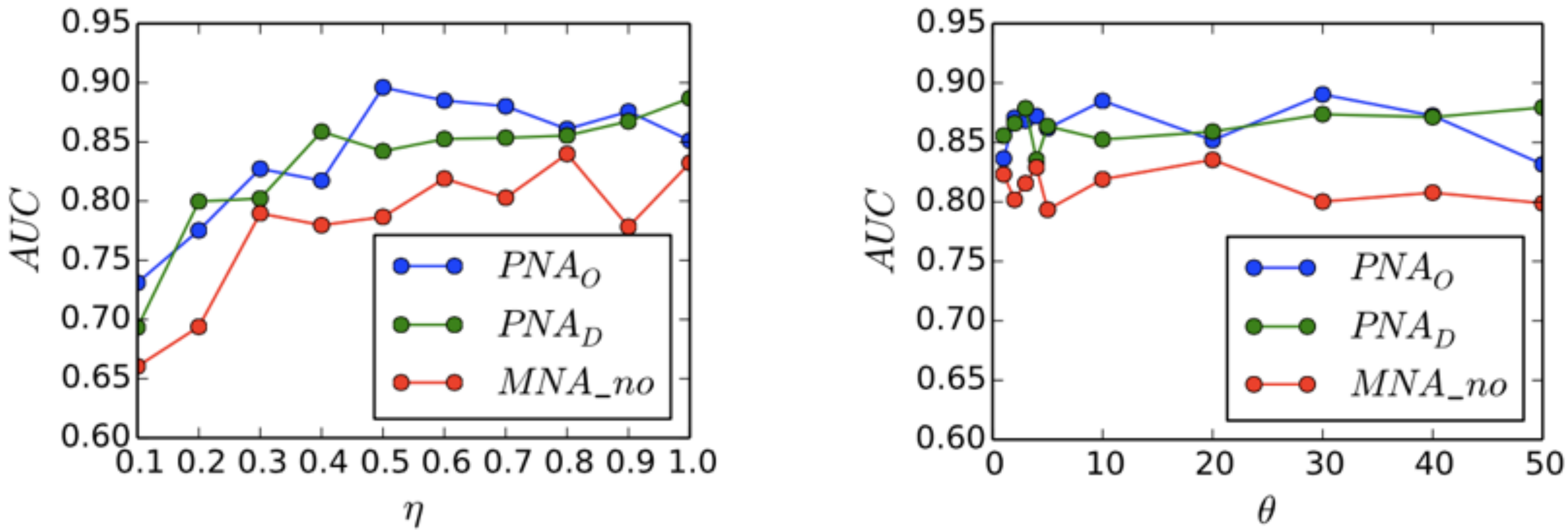
Experiment Settings

- Comparison Methods:
  - PNA_{OMG}: Link Prediction (Over Sampling) + Generic Stable Matching
  - PNA_{DMG}: Link Prediction (Down Sampling) + Generic Stable Matching
  - PNA_{OM}: Link Prediction (Over Sampling) + Traditional Stable Matching
  - PNA_{DM}: Link Prediction (Down Sampling) + Traditional Stable Matching
  - PNA_{O}: Link Prediction (Over Sampling)
  - PNA_{D}: Link Prediction (Down Sampling)
  - MNA: Link Prediction without Sampling + Traditional Stable Matching
  - MNA-no: Link Prediction without Sampling

	PNA_{OMG}	PNA_{DMG}	PNA_{OM}	PNA_{DM}	PNA_{O}	PNA_{D}	MNA	MNA-no
over sampling	√		√		√			
down sampling		√		√		√		
generic stable matching	√	√						
traditional stable matching			√	√			√	

- Evaluation Metrics: Accuracy, AUC, F1

Effectiveness of Sampling Methods



(a) AUC: alignment rate

(b) AUC: neg. pos. rate

η: percentage of existing anchor links

θ: class imbalance rate, i.e., negative instance/positive instance

Remarks: PNA$_O$, PNA$_D$ and MNA-no are identical, except

- PNA$_O$ uses over sampling to hand class imbalance issue
- PNA$_D$ uses down sampling to deal with class imbalance problem
- MNA-no doesn't use any sampling methods at all

Observation:

PNA$_O$, PNA$_D$ can outperform MNA-no consistently for networks with different η and θ

Explanation:

Over sampling and down sampling works well in dealing with the class imbalance problem

Experiment Results

η: percentage of existing anchor links

	Methods	0.1	0.2	0.3	0.4	0.5	0.6	0.7
		anchor link sampling rate η						
Acc	PNAOMG	**0.964**	**0.966**	**0.973**	**0.967**	**0.987**	**0.989**	**0.981**
	PNADMG	**0.960**	**0.974**	**0.961**	**0.976**	**0.983**	**0.975**	**0.982**
	PNAOM	0.942	0.938	0.948	0.945	0.954	0.960	0.970
	PNADM	0.940	0.951	0.949	0.929	0.949	0.947	0.969
	MNA	0.917	0.918	0.922	0.922	0.931	0.937	0.940
	PNAO	0.905	0.907	0.915	0.915	0.918	0.927	0.926
	PNAD	0.905	0.908	0.911	0.912	0.915	0.926	0.923
	MNA_no	0.895	0.899	0.901	0.907	0.916	0.921	0.922
F1	PNAOMG	**0.280**	**0.375**	**0.442**	**0.496**	**0.615**	**0.717**	**0.776**
	PNADMG	**0.283**	**0.374**	**0.412**	**0.481**	**0.589**	**0.658**	**0.783**
	PNAOM	0.230	0.318	0.384	0.452	0.543	0.638	0.723
	PNADM	0.239	0.324	0.369	0.424	0.526	0.593	0.716
	MNA	0.211	0.267	0.375	0.420	0.496	0.578	0.705
	PNAO	0.014	0.054	0.211	0.210	0.305	0.402	0.413
	PNAD	0.010	0.048	0.131	0.165	0.257	0.380	0.365
	MNA_no	0.004	0.021	0.042	0.067	0.232	0.322	0.339

in this table, class imbalance rate θ, i.e., negative/positive = 10

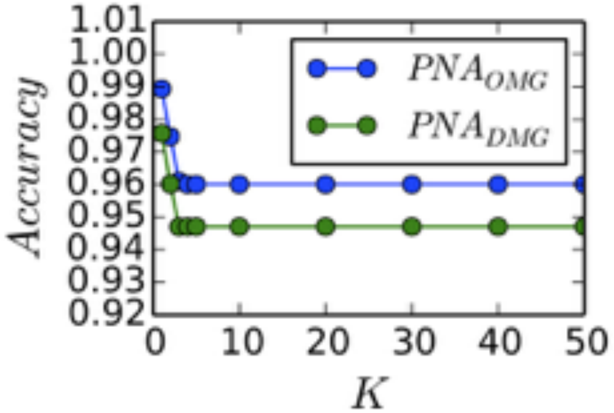
Observations:

1. All the methods achieve better results as η increases
2. Accuracy score achieved by all methods are very high
3. PNAOMG (PNADMG) performs better than PNAOM (PNADM)
4. PNAOM and PNADM achieves better results than MNA
5. PNAOM (PNADM and MNA) out-perform PNAO (PNAD and MNA-no)

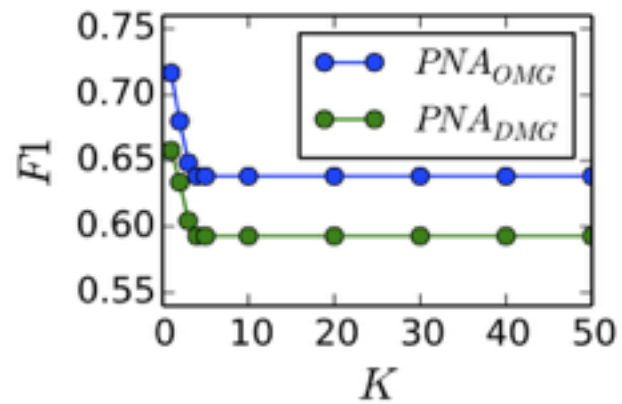
Explanations

1. more anchor links, more training instances to build models
2. due to the class imbalance problem, all these methods can make correct prediction of negative links easily and achieve high accuracy
3. generic stable matching and self matching works better for partial network alignment than traditional stable matching
4. over sampling and down sampling works well in addressing the class imbalance problem
5. stable matching is helpful for pruning non-existing anchor links

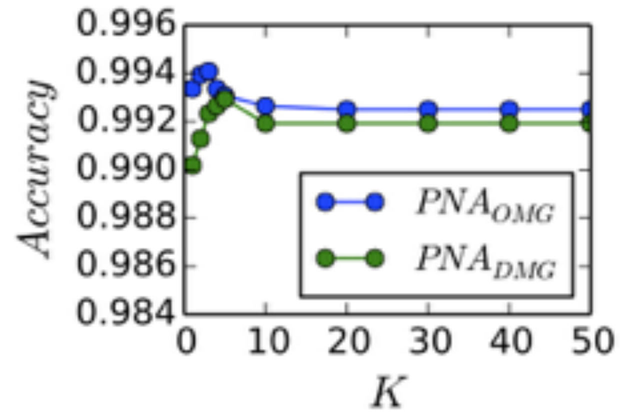
Parameter Analysis



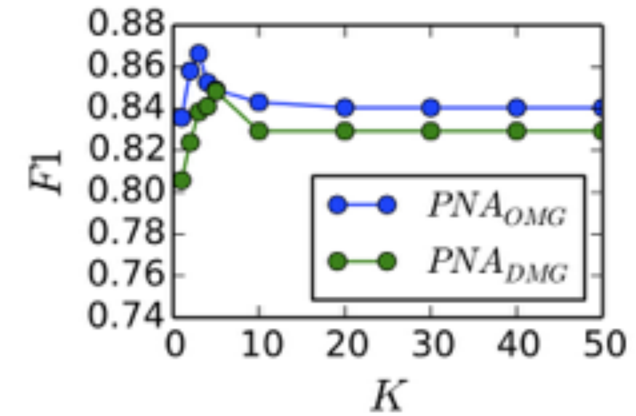
(a) Acc.@$\theta = 5$, $\eta = 0.4$



(b) F1@$\theta = 5$, $\eta = 0.4$



(c) Acc.@$\theta = 50$, $\eta = 0.9$



(d) F1@$\theta = 50$, $\eta = 0.9$

Observations: for networks with lower class imbalance rates and alignment rate (e.g., θ=5, η=0.4)

- the optimal “partial alignment rate” K for methods PNA_{OMG} and PNA_{DMG} is 1, i.e., the optimal matching results are candidates with the highest prediction scores
- performance of PNA_{OMG} and PNA_{DMG} will become worse as K increases from1 to 5
- as K further increases, it will have no effects on PNA_{OMG} and PNA_{DMG}, as candidates which are far behind in the preference list will never be selected in the matching result

Observations: for networks with higher class imbalance rates and alignment rate (e.g., θ=50, η=0.9)

- the optimal “partial alignment rate” K for methods PNA_{OMG} and PNA_{DMG} are 3 and 5 respectively,
- performance of PNA_{OMG} (PNA_{DMG}) will become worse as K increases from1 to 3 (1 to 5), but drops are K increases to 10
- as K further increases, it will have no effects on PNA_{OMG} and PNA_{DMG}, as candidates which are far behind in the preference list will never be selected in the matching result

Summary

- In this paper, we study the partial network alignment problem.
- A 2-phrase network alignment framework, PNA, is introduced to address the problem
  - step 1: supervised anchor link prediction
    - over sampling/down sampling to handle the class imbalance problem
    - extract features from across the heterogeneous networks based on a set of anchor meta paths
  - step 2: partial network matching with Generic Stable Matching to maintain the 1-to-1≤ constraint on anchor links
    - self matching is introduced to deal with unshared users

PNA: Partial Network Alignment with Generic Stable Matching

Q&A

Jiawei Zhang[1], Weixiang Shao[1], Senzhang Wang[2],
Xiangnan Kong[3], and Philip S. Yu[1,4]
jzhan9@uic.edu,wshao4@uic.edu, szwang@cse.buaa.edu.cn, xkong@wpi.edu,
psyu@cs.uic.edu