Graph Mining & Multi-Relational Learning
Tools and Applications
Part II

Shobeir Fakhraei
Amazon

Christos Faloutsos
CMU / Amazon

Bird's eye view

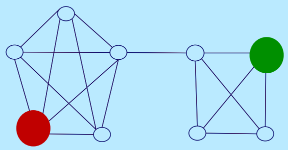
Tool / Task	1.1 PR/HITS	1.1 PPR	1.2 METIS/ SVD	1.3 OddBall+	1.4 BP	2.1 FM	2.1 Tensor	2.2 HIN	2.3 SRL
1.1 Node Ranking	👍								
1.1' Link Prediction		👍							
1.2 Comm. Detection			👍						
1.3 Anomaly Detection				👍					
1.4 Propagation					👍				

Part 1: Plain Graphs

Part 2: Complex Graphs

Bird's eye view

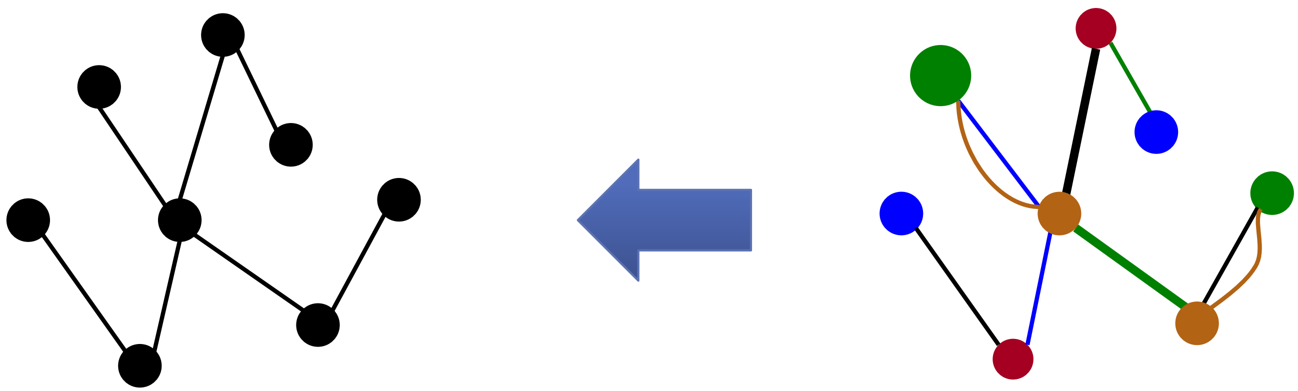
- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
  - P 2.2: Heterogeneous Information Networks
  - P 3.3: Statistical Relational Learning



Complex Networks

What Plain Graphs Tools Capture

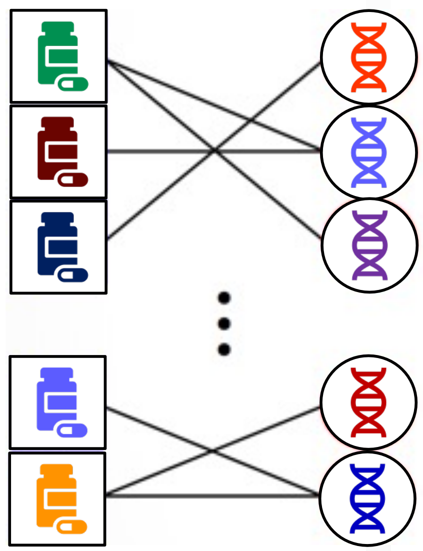
Complex Networks

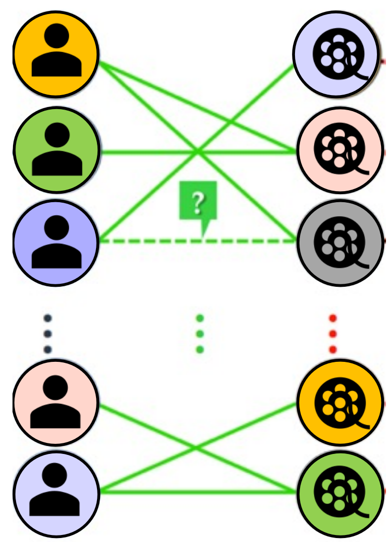


Complex Network (Many Related Terms)

- Plain Graphs + Extra Information on?
  - Nodes?
    - Multi-typed Networks
  - Edges?
    - Multi-layer Networks
    - Multi-dimensional Networks
    - Multi-modal Networks
  - Both?
    - Attributed Networks
    - Multiplex Networks
    - Multi Relational Networks
    - Heterogenous Information Networks
    - Complex Networks
  - ...

Common Bipartite Structure


Drug-Target Interactions



Recommender Systems

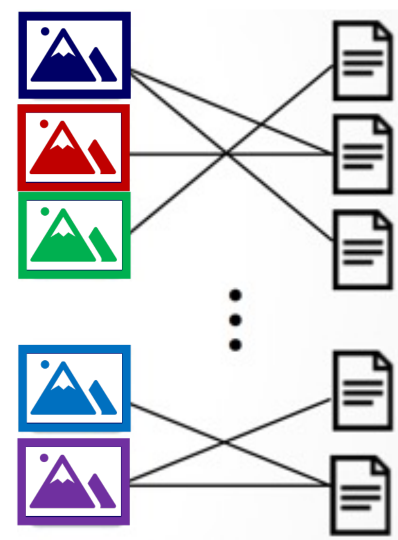
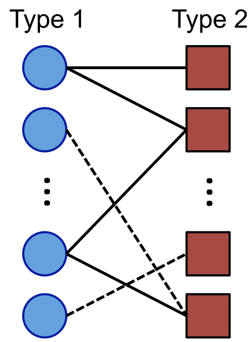


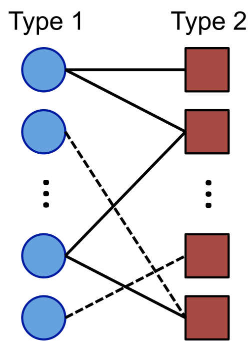
Image Captioning

Complex Bipartite Network

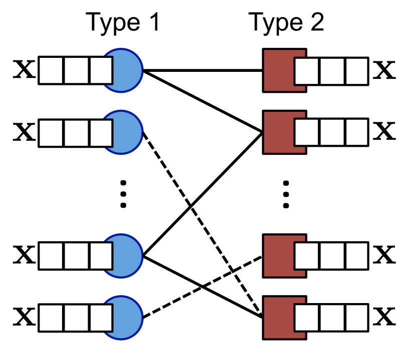


Two types of nodes
and relation of interest

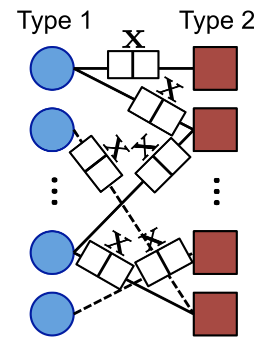
Complex Bipartite Network



Two types of nodes
and relation of interest

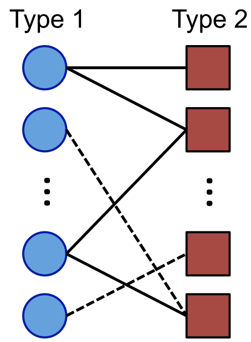


Additional features
for nodes

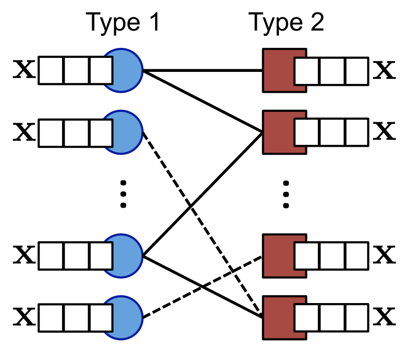


Additional features
for the relation

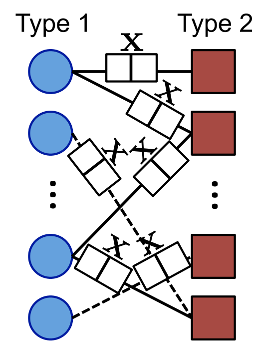
Complex Bipartite Network



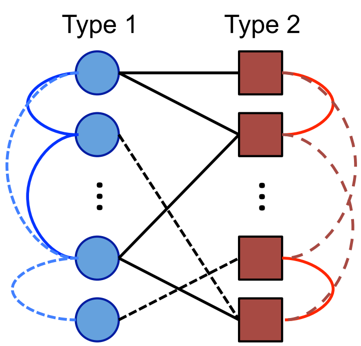
Two types of nodes
and relation of interest



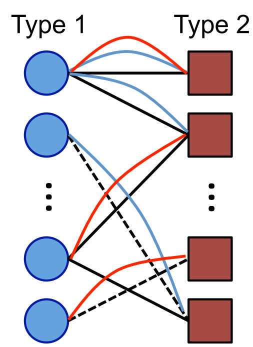
Additional features
for nodes



Additional features
for the relation

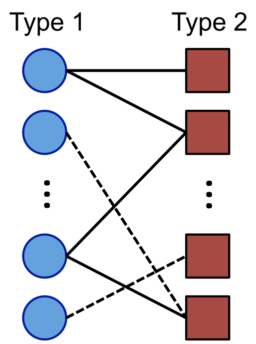


Additional relations
for nodes

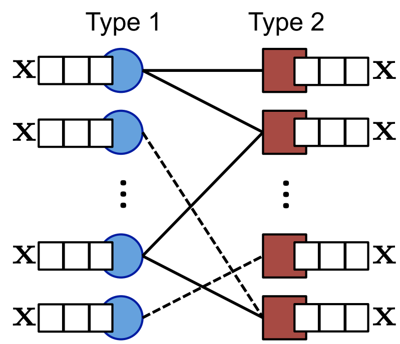


Additional relations
for the relation

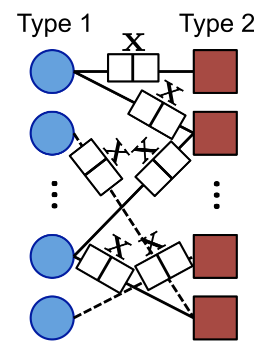
Complex Bipartite Network



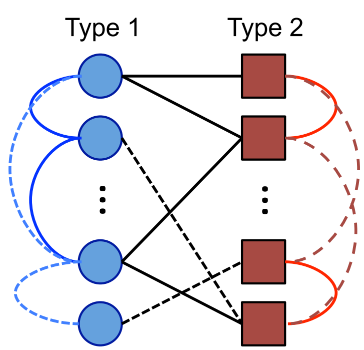
Two types of nodes and relation of interest



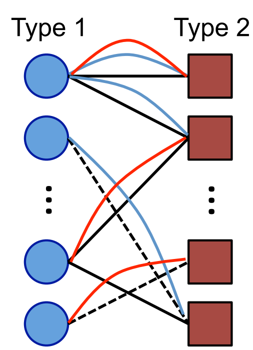
Additional features for nodes



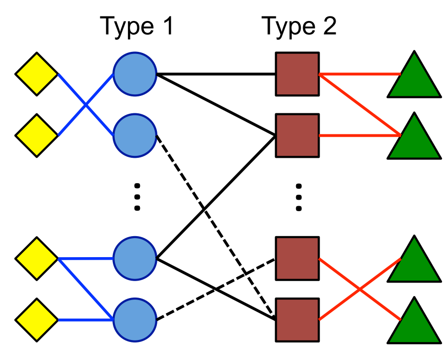
Additional features for the relation



Additional relations for nodes

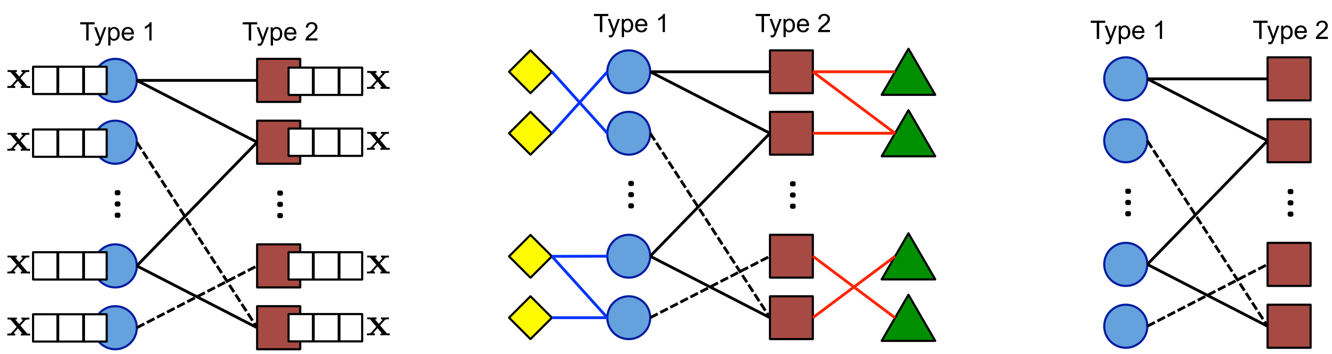


Additional relations for the relation

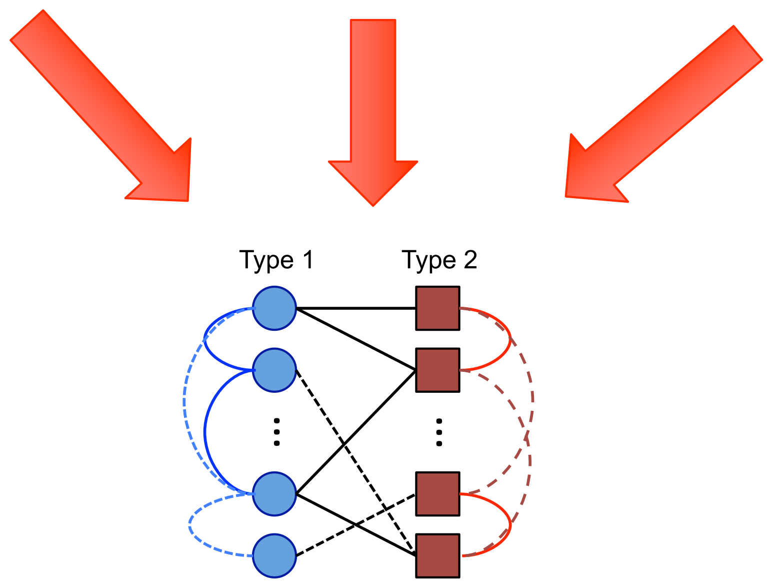


Additional relations with external nodes

Complex Bipartite Network

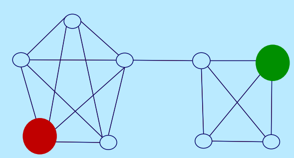


Euclidean Distance Jaccard/Cosine Similarity



Bird's eye view

- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
    - P 2.1.1: Factorization Machines
    - P 2.1.2: Tensor Methods
  - P 2.2: Heterogeneous Information Networks
  - P 3.3: Statistical Relational Learning

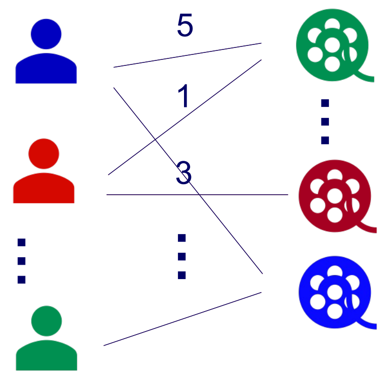


Question:

- Q: How can we add extra information to a bipartite-graph for link predication / recommender systems?

- A: Factorization Machines is one way!

Bipartite Graph

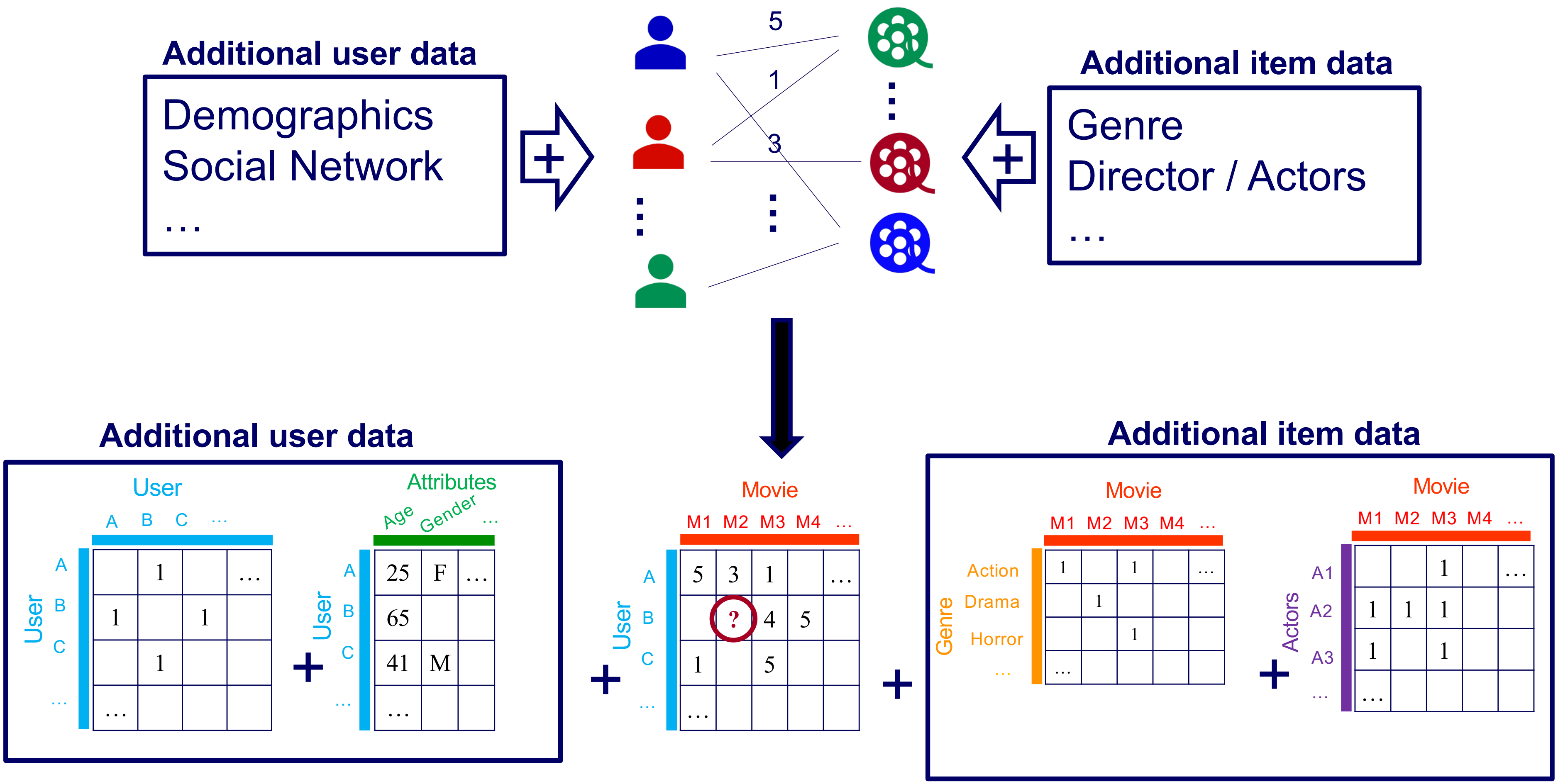


Movie

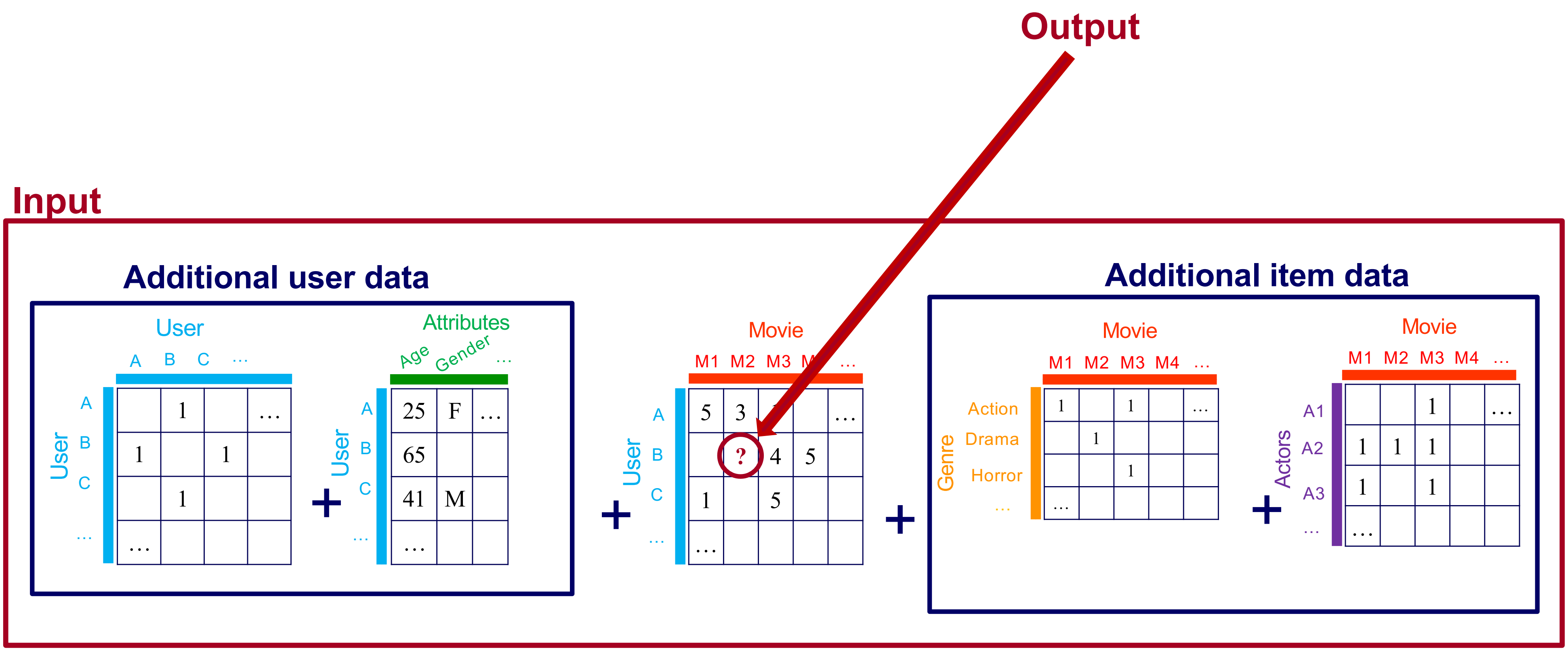
User		M1	M2	M3	M4	…
	A	5	3	1		…
	B		?	4	5	
	C	1		5		
	…	…				

→ SVD / Matrix Factorization

How to include additional data?



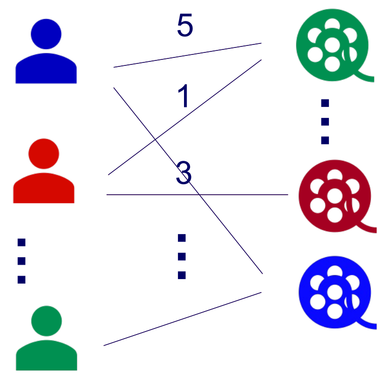
How to include additional data?



How to include additional data?

Additional user data

Demographics
Social Network
…



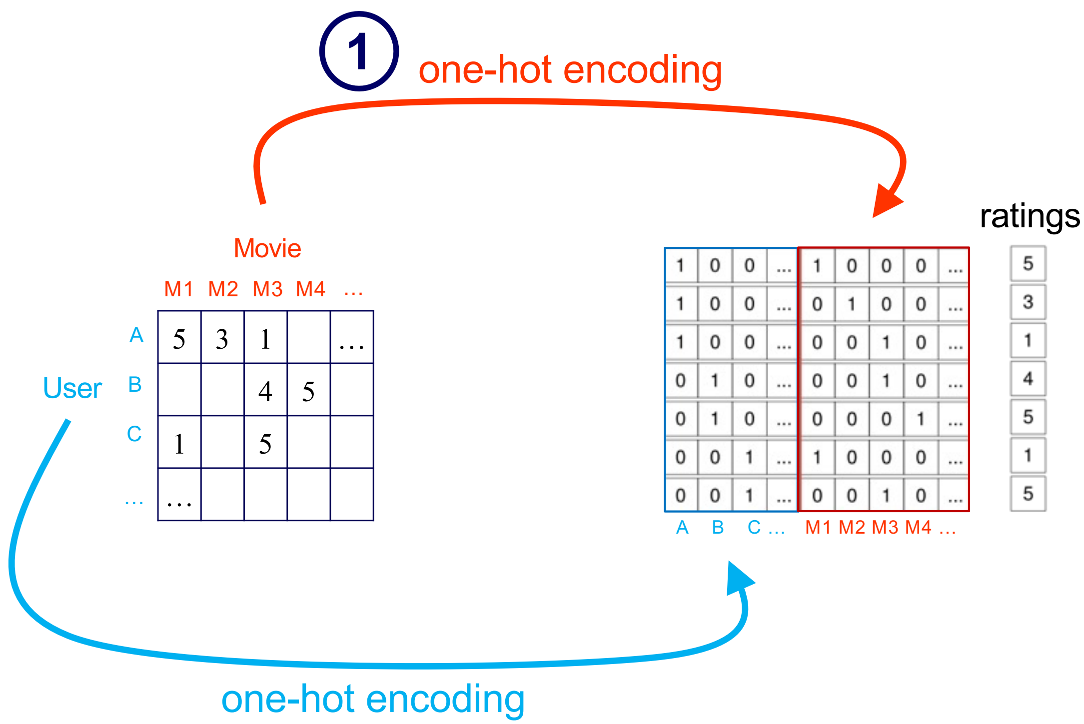
Additional item data

Genre
Director / Actors
…

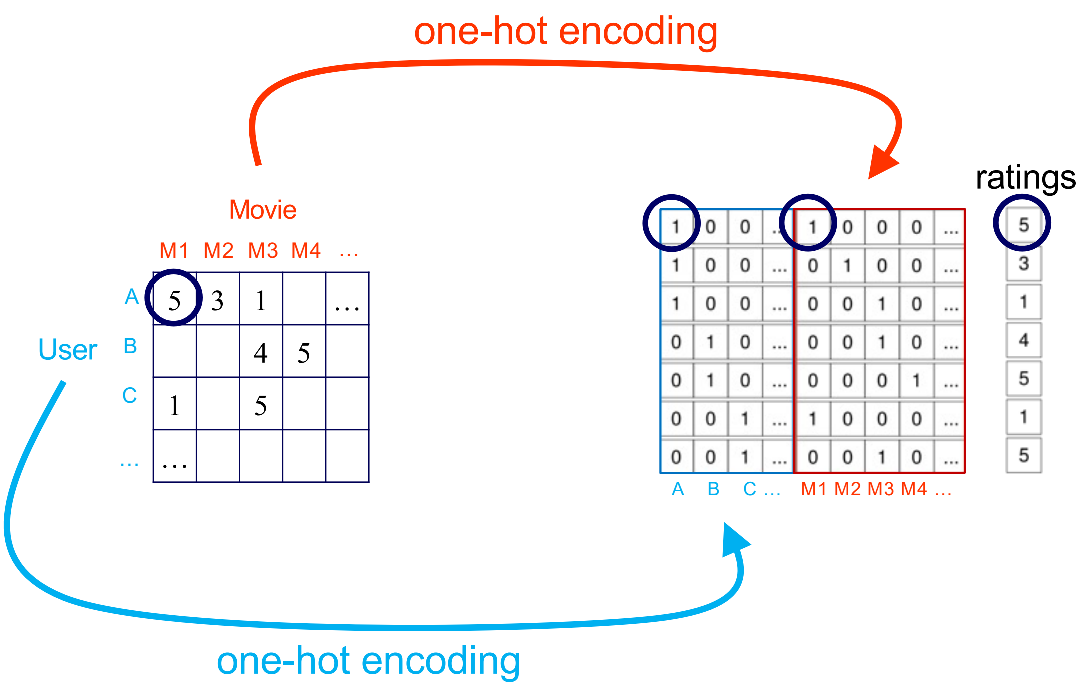
One answer: Factorization Machines

1. One-hot encoding
2. Pairwise Interactions
3. Latent factor representation

Data Representation in FM



Data Representation in FM



Factorization Machines

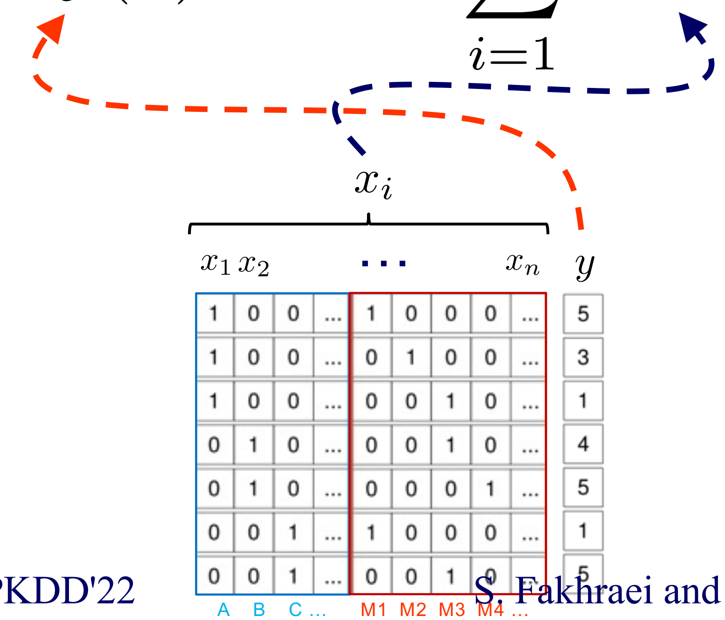
$$\overbrace{x_1\ x_2\ \cdots\ x_n}^{x_i} \quad y$$

A	B	C ...		M1	M2	M3	M4 ...		y
1	0	0	...	1	0	0	0	...	5
1	0	0	...	0	1	0	0	...	3
1	0	0	...	0	0	1	0	...	1
0	1	0	...	0	0	1	0	...	4
0	1	0	...	0	0	0	1	...	5
0	0	1	...	1	0	0	0	...	1
0	0	1	...	0	0	1	0		5

Details

Factorization Machines

$$\hat{y}(\mathbf{x}) = \overbrace{w_0}^{\text{Global bias}} + \sum_{i=1}^{n} \overbrace{w_i}^{\text{Regression coefficient of i-th variable.}} x_i$$



1	0	0	...	1	0	0	0	...	5
1	0	0	...	0	1	0	0	...	3
1	0	0	...	0	0	1	0	...	1
0	1	0	...	0	0	1	0	...	4
0	1	0	...	0	0	0	1	...	5
0	0	1	...	1	0	0	0	...	1
0	0	1	...	0	0	1	0	...	5

Details

Factorization Machines

Linear Regression

$$\hat{y}(\mathbf{x}) = \overbrace{w_0}^{\text{Global bias}} + \sum_{i=1}^{n} \overbrace{w_i}^{\text{Regression coefficient of i-th variable.}} x_i$$

x_i

x_1	x_2		...					x_n	y
1	0	0	...	1	0	0	0	...	5
1	0	0	...	0	1	0	0	...	3
1	0	0	...	0	0	1	0	...	1
0	1	0	...	0	0	1	0	...	4
0	1	0	...	0	0	0	1	...	5
0	0	1	...	1	0	0	0	...	1
0	0	1	...	0	0	1	0	...	5
A	B	C ...		M1	M2	M3	M4 ...		

Details

Factorization Machines

Regression coefficient
of i-th variable.

Global bias

Pairwise interactions (2)

$$\hat{y}(\mathbf{x}) = \overbrace{w_0}^{\text{Global bias}} + \sum_{i=1}^{n} \overbrace{w_i}^{\text{Regression coefficient of i-th variable.}} x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \overbrace{w_{ij}}^{\text{Pairwise interactions}} x_i x_j$$

(1)

$\overbrace{x_1 \; x_2 \; \cdots \; x_n}^{x_i}$ y

A	B	C ...		M1	M2	M3	M4 ...		y
1	0	0	...	1	0	0	0	...	5
1	0	0	...	0	1	0	0	...	3
1	0	0	...	0	0	1	0	...	1
0	1	0	...	0	0	1	0	...	4
0	1	0	...	0	0	0	1	...	5
0	0	1	...	1	0	0	0	...	1
0	0	1	...	0	0	1	0	...	5

Details

Factorization Machines

Regression coefficient
of i-th variable.

Global bias

Pairwise interactions (2)

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

Impractical
to compute

(1) x_i

x_1	x_2	...						x_n	y
1	0	0	...	1	0	0	0	...	5
1	0	0	...	0	1	0	0	...	3
1	0	0	...	0	0	1	0	...	1
0	1	0	...	0	0	1	0	...	4
0	1	0	...	0	0	0	1	...	5
0	0	1	...	1	0	0	0	...	1
0	0	1	...	0	0	1	0	...	5
A	B	C ...		M1	M2	M3	M4 ...		

Factorization Machines

Details

$$\hat{y}(\mathbf{x}) = \overbrace{w_0}^{\text{Global bias}} + \sum_{i=1}^{n} \overbrace{w_i}^{\text{Regression coefficient of i-th variable.}} x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \overbrace{w_{ij}}^{\text{Pairwise interactions}} x_i x_j$$

2

$$w_{ij} \approx \hat{w}_{ij} = \langle \underbrace{\mathbf{v}_{\mathbf{x}_i}, \mathbf{v}_{\mathbf{x}_j}}_{\text{Latent factor for each column}} \rangle$$

3

1

x_i

x_1	x_2		...					x_n	y
1	0	0	...	1	0	0	0	...	5
1	0	0	...	0	1	0	0	...	3
1	0	0	...	0	0	1	0	...	1
0	1	0	...	0	0	1	0	...	4
0	1	0	...	0	0	0	1	...	5
0	0	1	...	1	0	0	0	...	1
0	0	1	...	0	0	1	0	...	5
A	B	C ...		M1	M2	M3	M4 ...		

Details

Factorization Machines

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \overbrace{\langle \mathbf{v}_{\mathbf{x}_i}, \mathbf{v}_{\mathbf{x}_j} \rangle}^{w_{ij}} x_i x_j$$

Regression $\Rightarrow$ $$\min \sum_{E} (y - \hat{y})^2 + \lambda_1 ||\mathbf{w}_i||^2 + \lambda_2 ||\mathbf{V}_{\mathbf{x}}||^2$$

X

User				Movie					Time					y
A	B	C	...	M1	M2	M3	M4	...	1am	2am	3am	4am	...	
1	0	0	...	1	0	0	0	...	0	0	0	0	...	5
1	0	0	...	0	1	0	0	...	1	0	0	0	...	3
1	0	0	...	0	0	1	0	...	0	1	0	0	...	1
0	1	0	...	0	0	1	0	...	0	0	0	0	...	4
0	1	0	...	0	0	0	1	...	0	0	1	0	...	5
0	0	1	...	1	0	0	0	...	0	0	0	0	...	1
0	0	1	...	0	0	1	0	...	1	0	0	0	...	5

Data Representation in FM

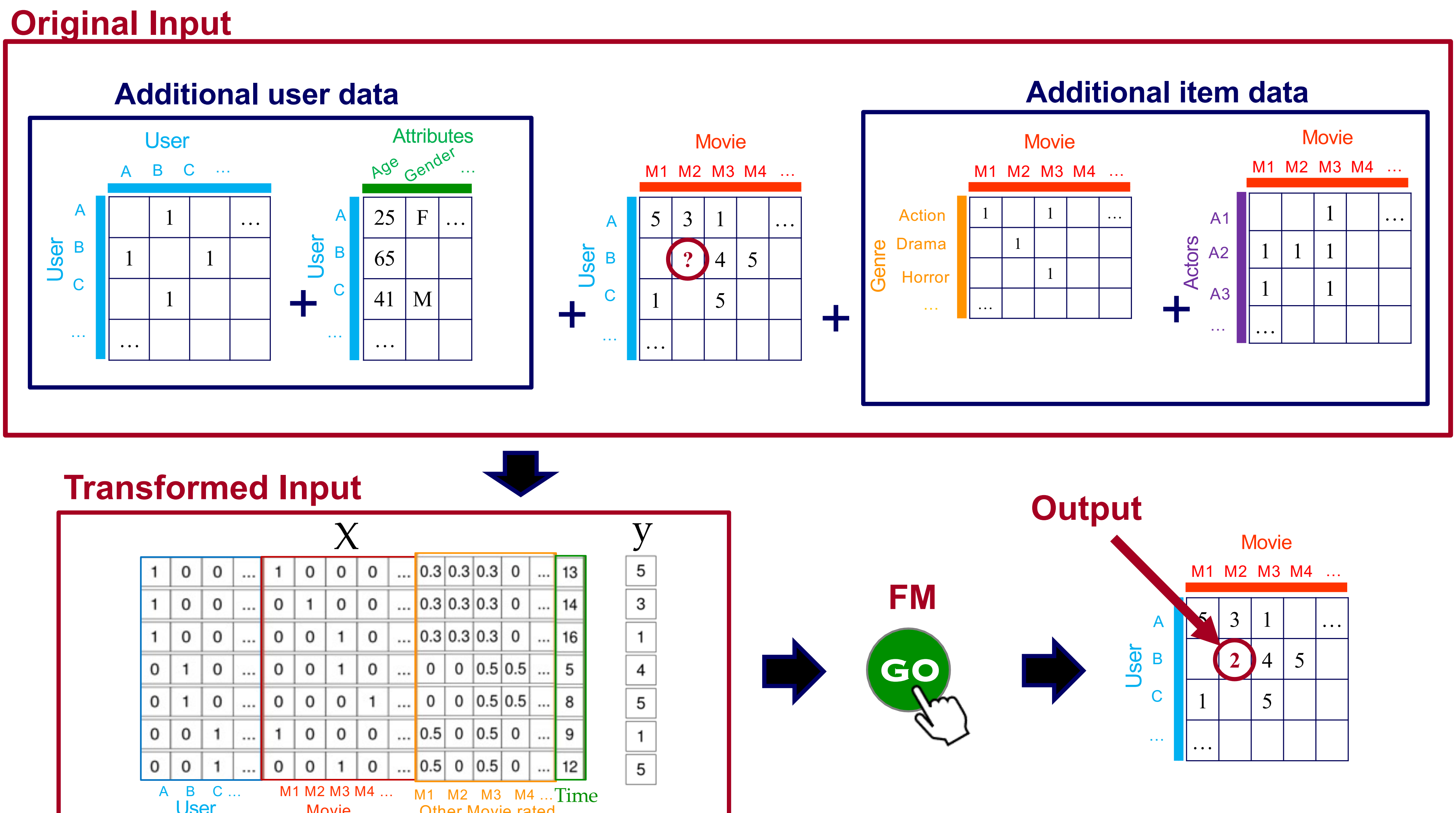
- Categorical Information (One-hot encoding)
  e.g., User and item ID

- Set Information
  e.g., list of friends, other movies watched

- Continuous
  e.g., time, location, age

X															y
User				Movie					Other Movie rated					Time	
A	B	C	...	M1	M2	M3	M4	...	M1	M2	M3	M4	...		
1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	5
1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	3
1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	1
0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	4
0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	5
0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	1
0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	5

Take Away


Original Input
Additional user data
User
A B C …
User
A B C …
Attributes
Age Gender …
User
A B C …
25 F …
65
41 M
…
Movie
M1 M2 M3 M4 …
5 3 1 …
? 4 5
1 5
Additional item data
Movie
M1 M2 M3 M4 …
Genre
Action
Drama
Horror
…
Movie
M1 M2 M3 M4 …
Actors
A1
A2
A3
…
Transformed Input
X
y
User
A B C …
Movie
M1 M2 M3 M4 …
Other Movie rated
M1 M2 M3 M4 …
Time
FM
GO
Output
Movie
M1 M2 M3 M4 …
User
A B C …
5 3 1 …
2 4 5
1 5

Software Tools

– SageMaker Factorization Machines: https://docs.aws.amazon.com/sagemaker/latest/dg/fact-machines.html

– libFM: http://www.libfm.org/

References

- Rendle, Steffen
  Factorization machines with libfm
  ACM Transactions on Intelligent Systems and Technology (TIST), 2012

Bird's eye view

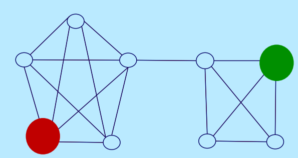
Task \ Tool	1.1 PR/HITS	1.1 PPR	1.2 METIS/ SVD	1.3 OddBall+	1.4 BP	2.1 FM	2.1 Tensor	2.2 HIN	2.3 SRL
1.1 Node Ranking*	👍					👍			
1.1' Link Prediction		👍				👍			
1.2 Comm. Detection			👍						
1.3 Anomaly Detection				👍					
1.4 Propagation					👍				

Part 1: Plain Graphs   **Part 2: Complex Graphs**

(* or Node Classification)

Bird's eye view

- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
    - P 2.1.1: Factorization Machines
    - P 2.1.2: Tensor Methods
  - P 2.2: Heterogeneous Information Networks
  - P 3.3: Statistical Relational Learning



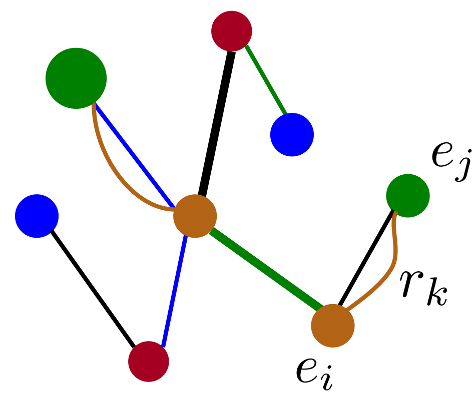
Question:

- Q: How can we add extra information to a graph and find communities?

- A: Tensors are one way!

Multi-relational network

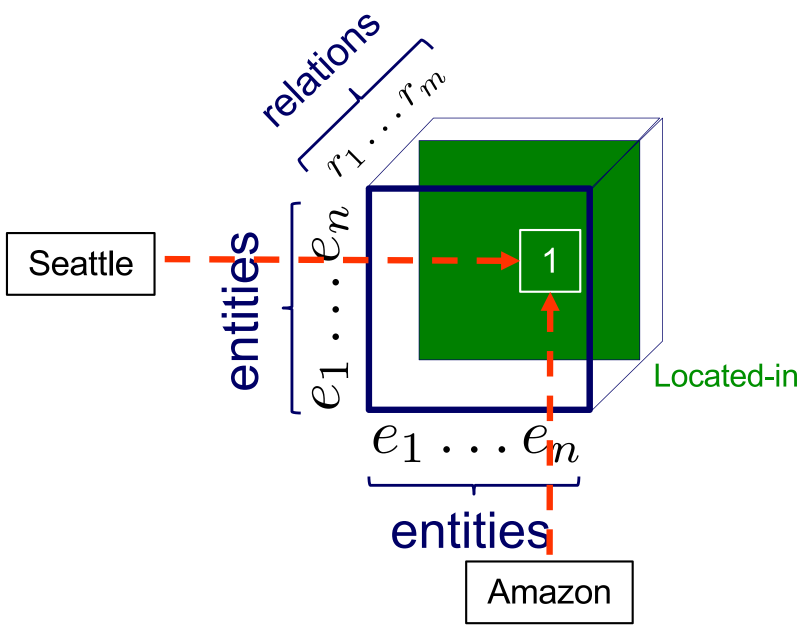
How to represent?



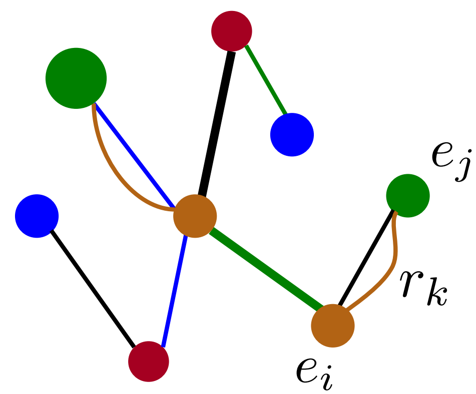
n: entities (e)
m: relations (r)

Example: Knowledge Graph

Multi-relational network



Tensor

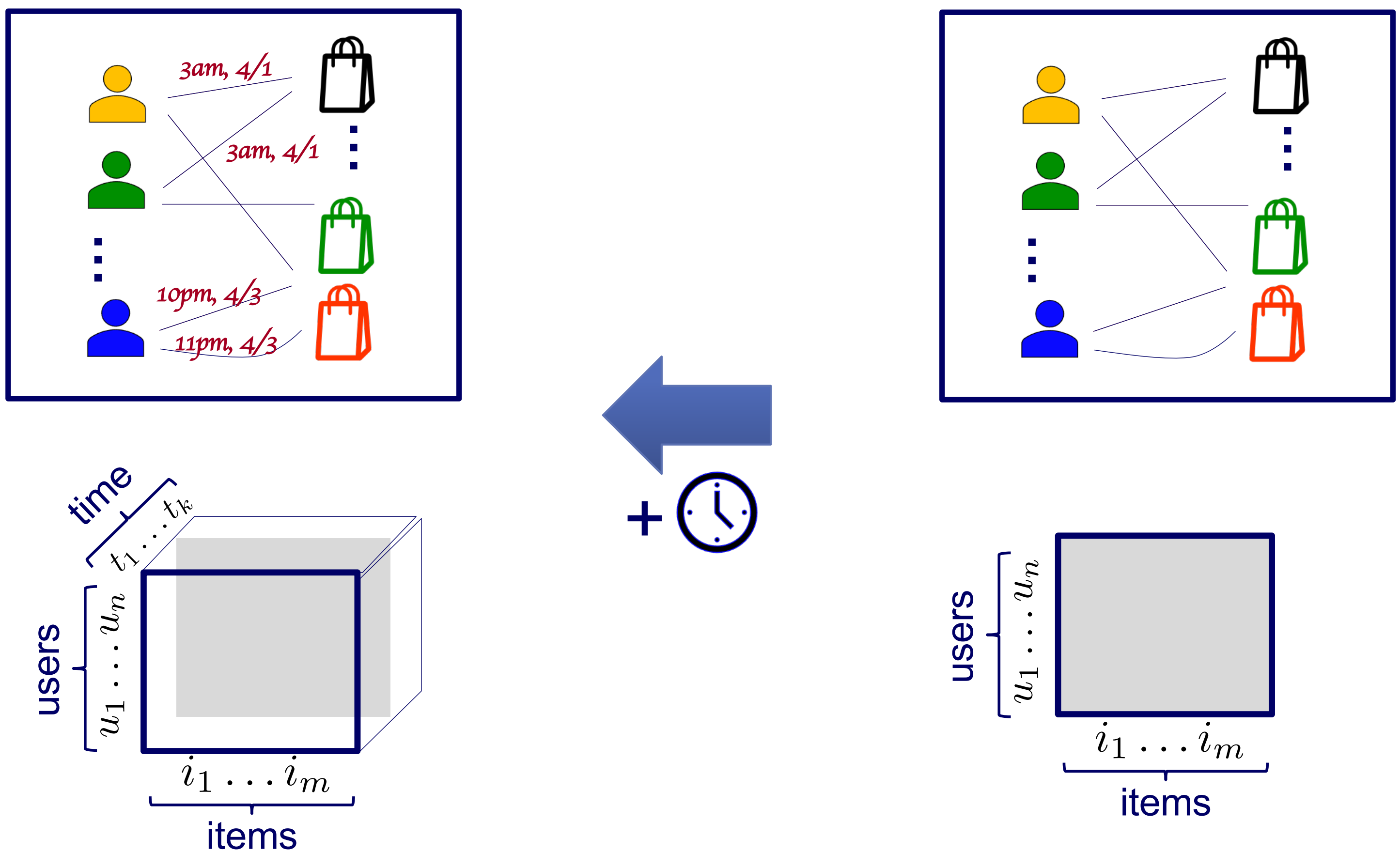


n: entities (e)
m: relations (r)

Time-evolving networks

who – buys – what - when

who – buys – what



Tensor examples

- Q: What is a tensor?
- A: N-D generalization of matrix:

KDD' 17	data	mining	classif.	tree	...
John	13	11	22	55	...
Peter	5	4	6	7	...
Mary	...	...	...	...	...
Nick	...	...	...	...	...
...	...	...	...	...	...

Tensor examples

- Q: What is a tensor?
- A: N-D generalization of matrix:

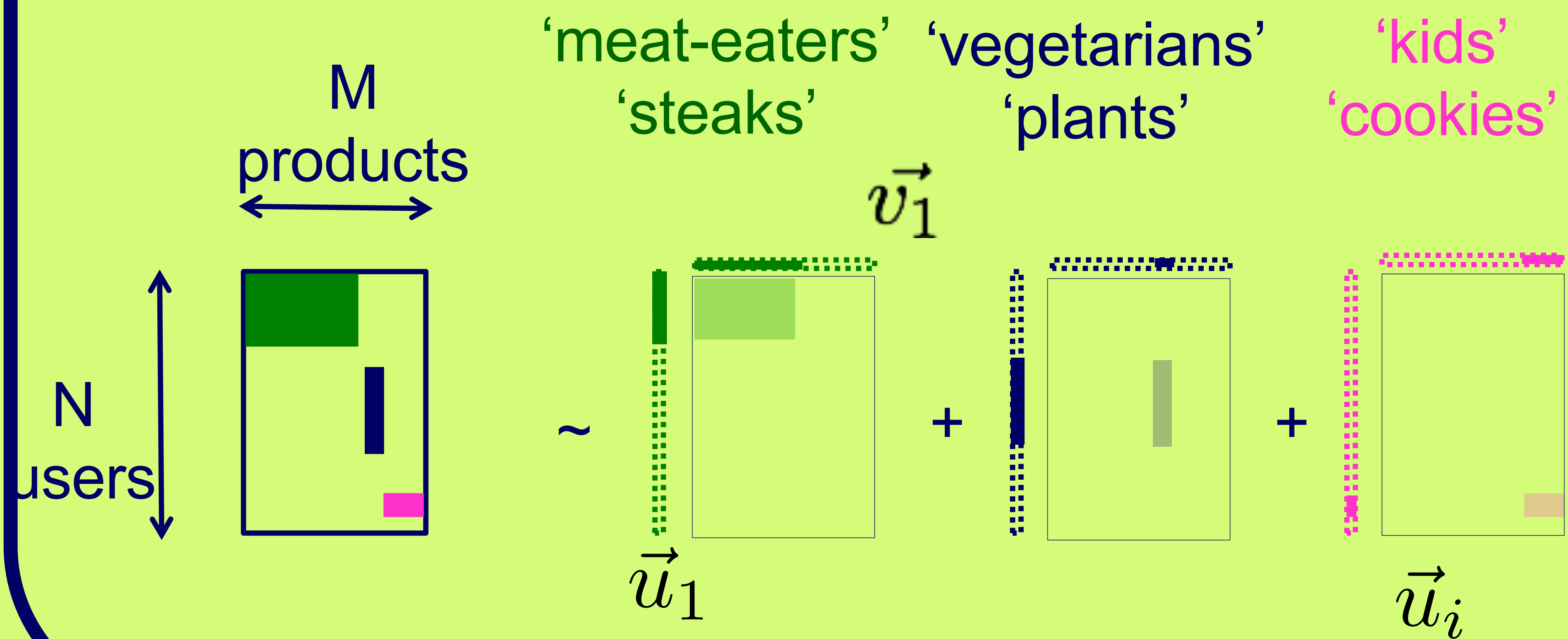
KDD' 19
KDD' 18
KDD' 17

	data	mining	classif.	tree	...
John	13	11	22	55	...
Peter	5	4	6	7	...
Mary	...	...	...	...	...
Nick	...	...	...	...	...
...	...	...	...	...	...

Tensor factorization

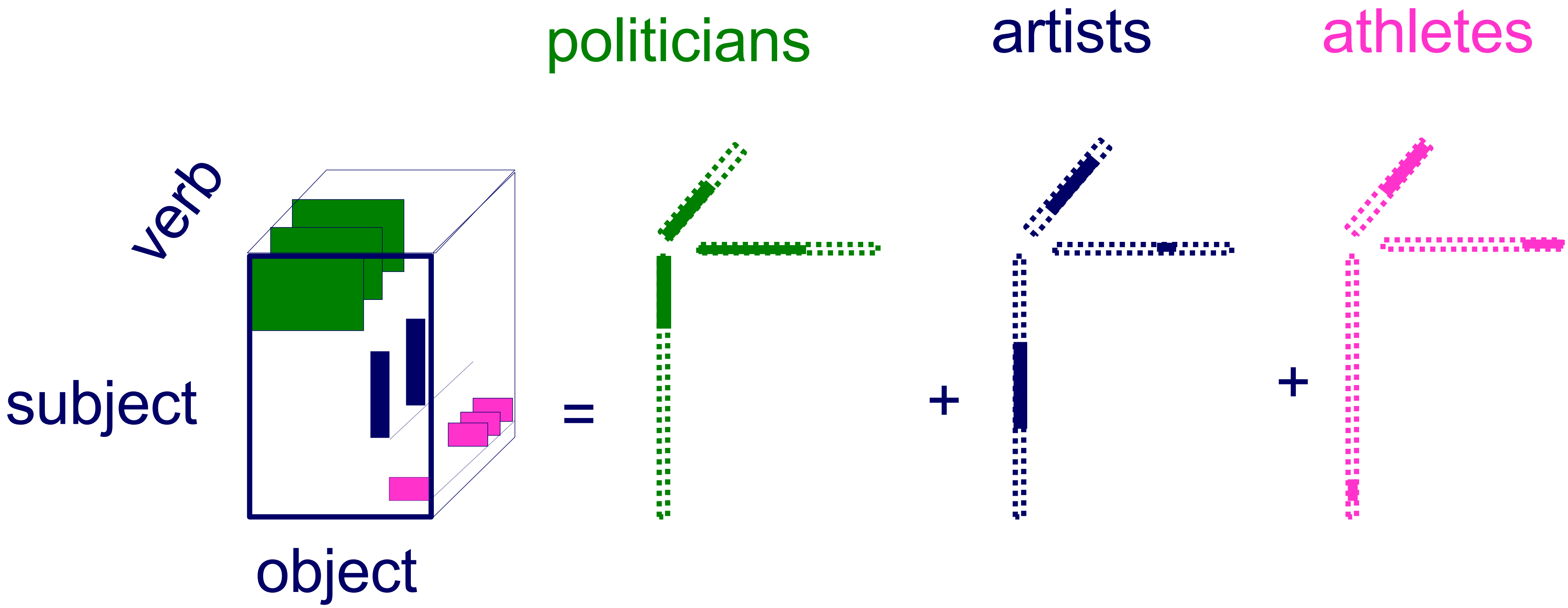
Reminder (from SVD)

- Recall: (SVD) matrix factorization: finds blocks



Tensor factorization

One Approach: PARAFAC decomposition

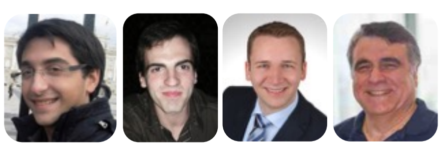


Example Applications

- TA1: Phonecall
- TA2: Network traffic

TA1: Anomaly detection in time-evolving graphs

- Anomalous communities in phone call data:
  - European country, 4M clients, data over 2 weeks



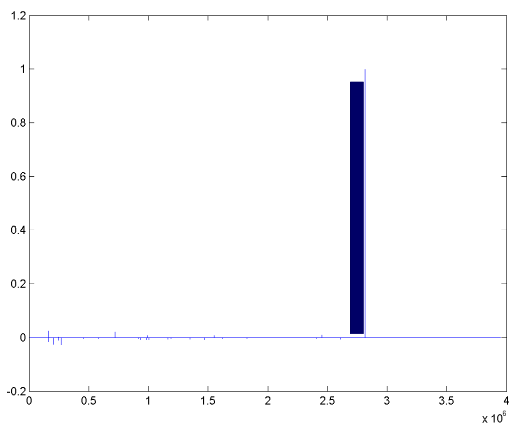
[PAKDD] "*Com2: Fast Automatic Discovery of Temporal (Comet) Communities",* Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami,Evangelos Papalexakis, Danai Koutra.

TA1: Anomaly detection in time-evolving graphs

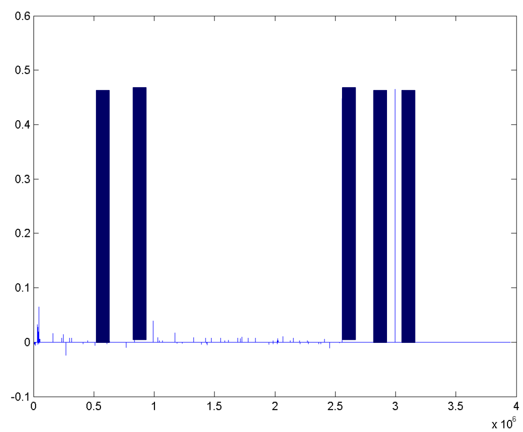
- Anomalous communities in phone call data:
  - European country, 4M clients, data over 2 weeks

PARAFAC

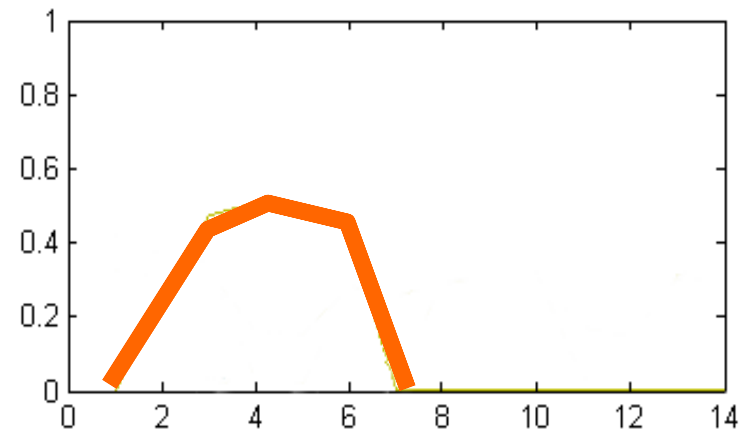
1 caller



5 receivers



4 days of activity



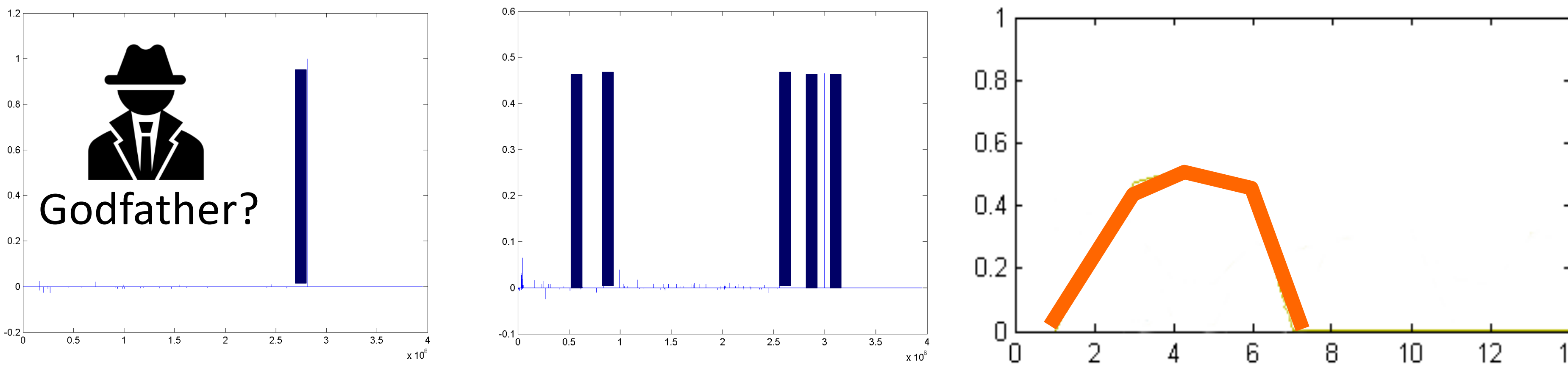
~200 calls to EACH receiver on EACH day!

TA1: Anomaly detection in time-evolving graphs

- Anomalous communities in phone call data:
  - European country, 4M clients, data over 2 weeks

PARAFAC

1 caller | 5 receivers | 4 days of activity



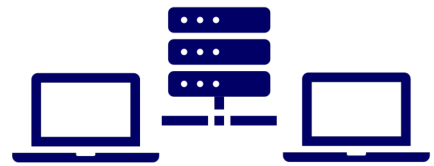
~200 calls to EACH receiver on EACH day!

Example Applications

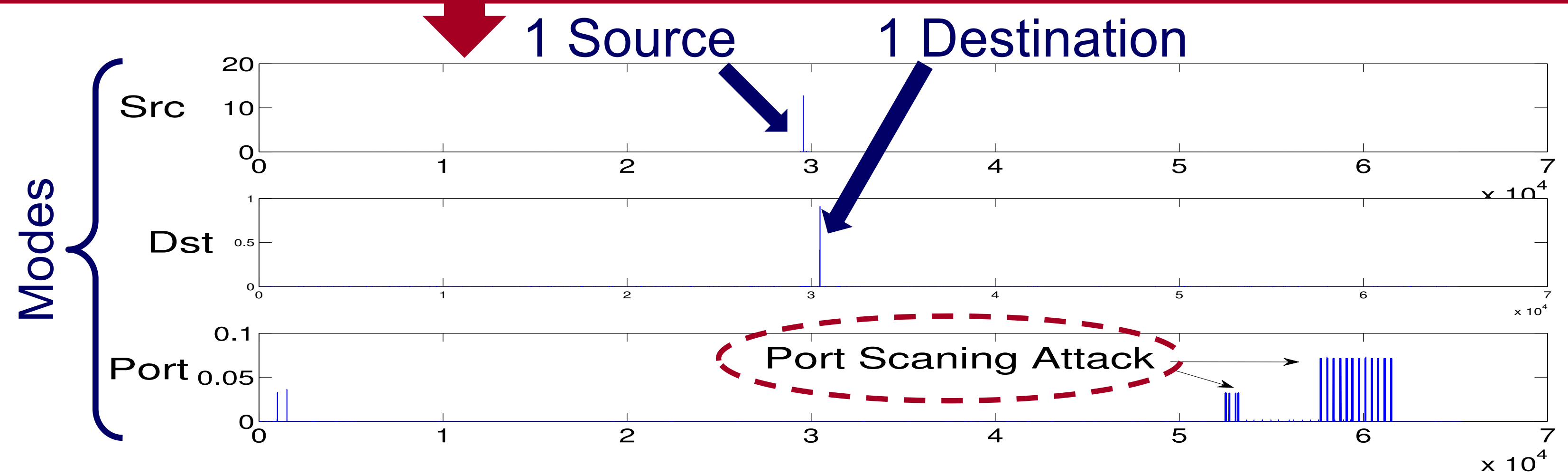
- TA1: Phonecall
- TA2: Network traffic

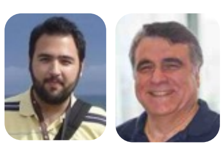
TA2: Anomaly detection in network traffic

BERKELEY LAB
Lawrence Berkeley National Laboratory



PARAFAC

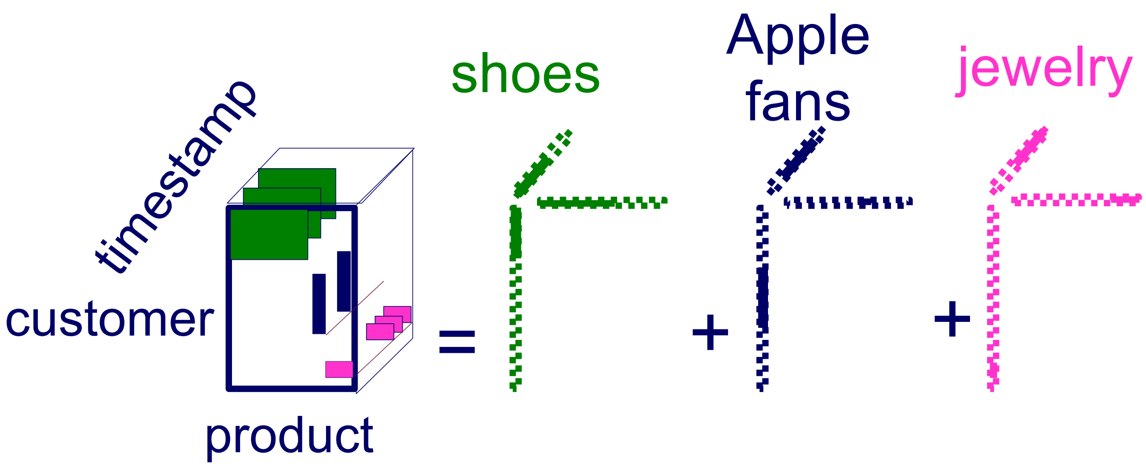




[ECML/PKDD] "ParCube: Sparse Parallelizable Tensor Decompositions", Evangelos E. Papalexakis, Christos Faloutsos, Nikos Sidiropoulos

Take Away

- Tensor analysis finds latent variables (e.g., market-segments)
  - Deviations → Anomalies
  - Link Prediction
- Extends SVD/factorization, to higher-modes



Software Tools

– TensorLy: Tensor Learning in Python
  http://tensorly.org/stable/index.html

– Tensor Toolbox for MATLAB
  http://www.tensortoolbox.org/

References

- Tamara G. Kolda and Brett W. Bader
  Tensor Decompositions and Applications
  SIAM Rev., 51(3), pp 455–500, 2009

- Nicholas D. Sidiropoulos, Lieven De Lathauwer,,
  Xiao Fu,, Kejun Huang, Evangelos E. Papalexakis,
  and Christos Faloutsos
  Tensor Decomposition for Signal Processing and Machine Learning
  IEEE TSP, 65(13), July 1, 2017

Bird's eye view

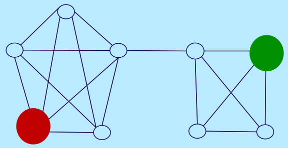
Task \ Tool	1.1 PR/HITS	1.1 PPR	1.2 METIS/ SVD	1.3 OddBall+	1.4 BP	2.1 FM	2.1 Tensor	2.2 HIN	2.3 SRL
1.1 Node Ranking	👍					👍			
1.1' Link Prediction		👍				👍	👍		
1.2 Comm. Detection			👍				👍		
1.3 Anomaly Detection				👍			👍		
1.4 Propagation					👍				

Part 1: Plain Graphs

Part 2: Complex Graphs

Bird's eye view

- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
  - P 2.2: Heterogeneous Information Networks
    - Metapaths
    - PathSim
  - P 3.3: Statistical Relational Learning

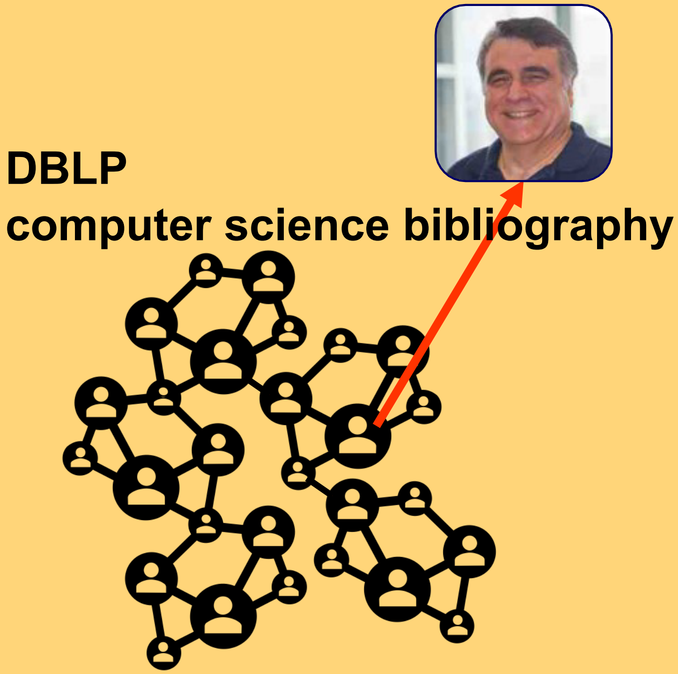


Question:

Q: How can we find node similarities in networks with extra information?

Question:

Q: In DBLP who are most similar to "Christos Faloutsos"?

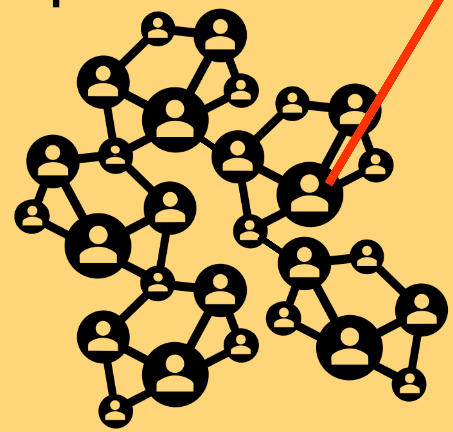


Question:

Q: In DBLP who are most similar to "Christos Faloutsos"?

How to define?

DBLP
computer science bibliography

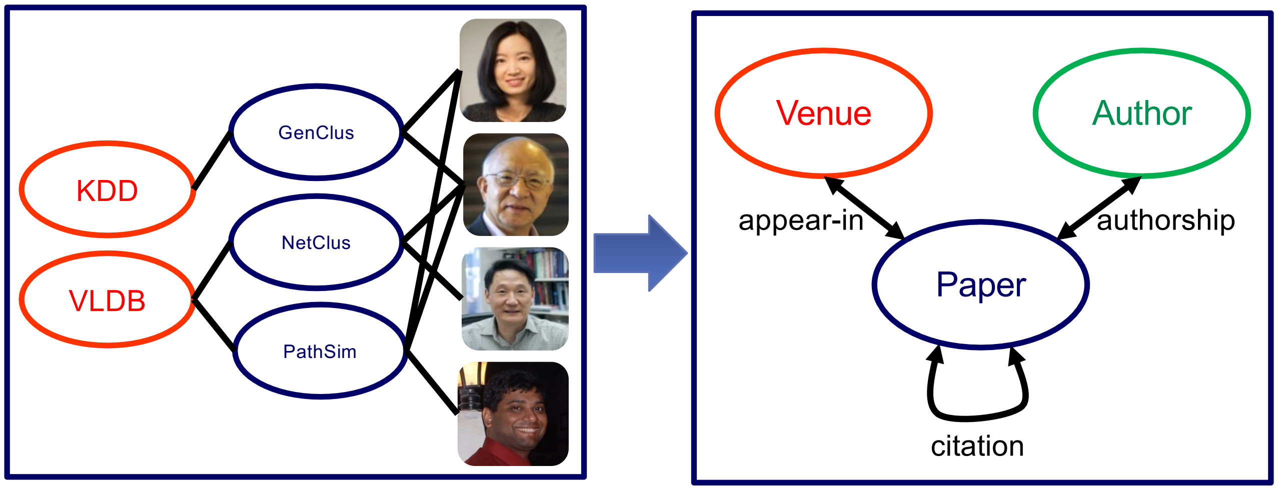


A: PathSim and Meta-path is one way!

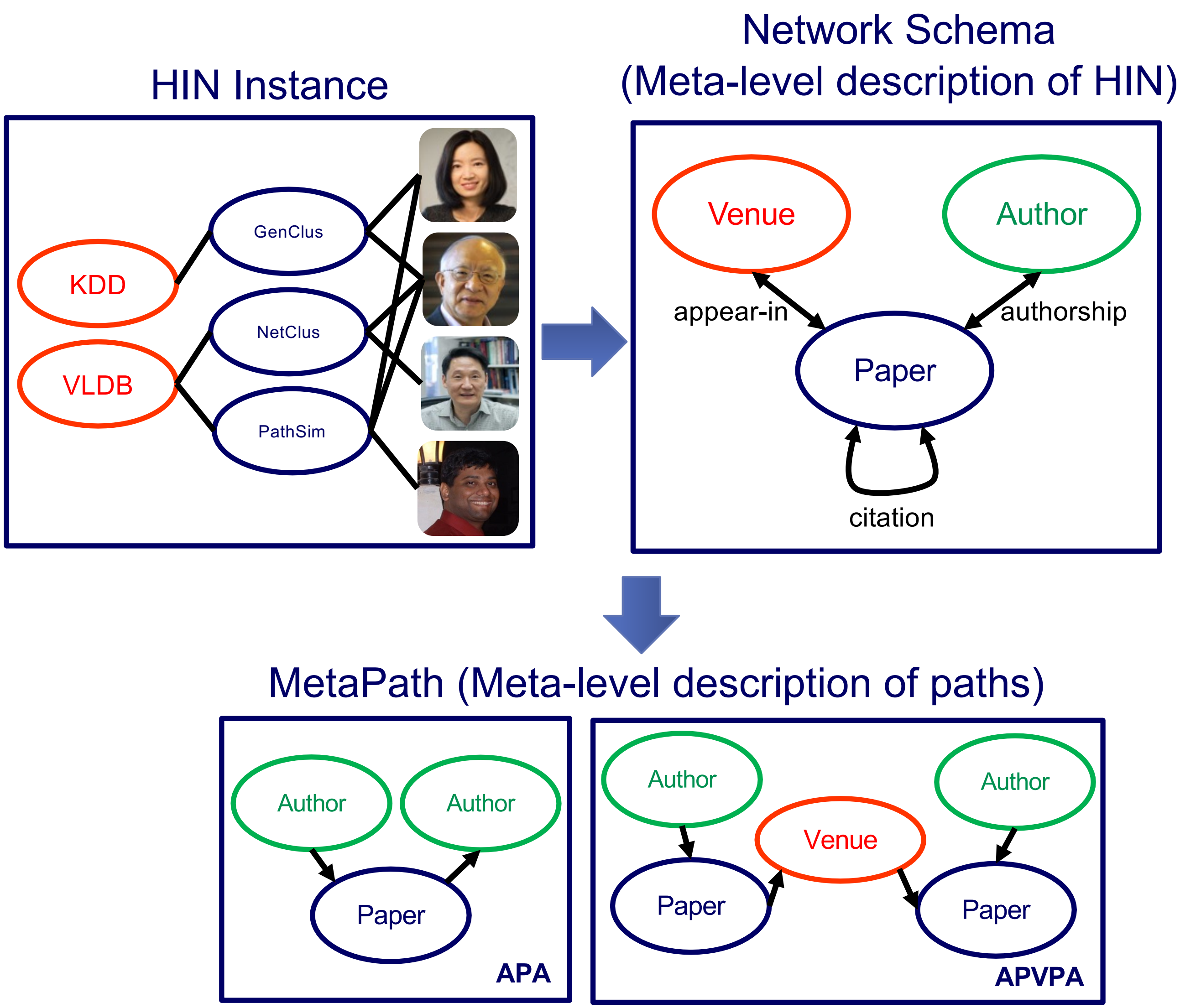
Heterogeneous Information Networks (HIN)

HIN Instance

Network Schema
(Meta-level description of HIN)

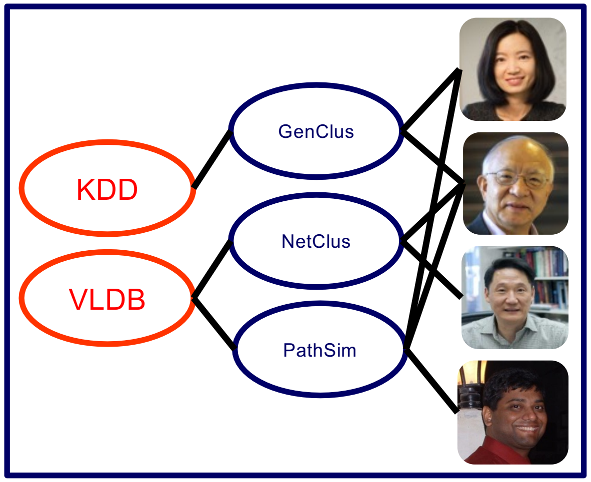


Heterogeneous Information Networks (HIN)

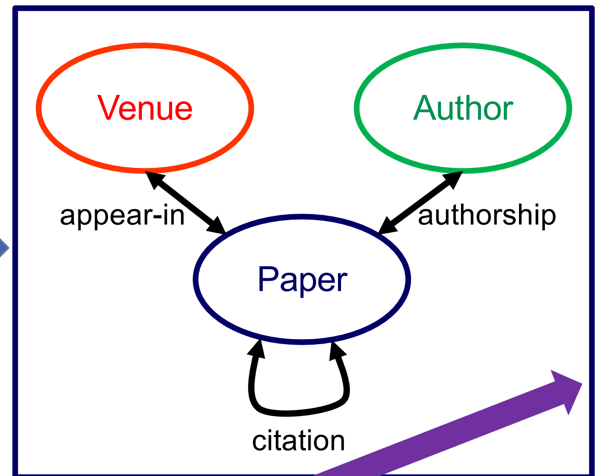


Heterogeneous Information Networks (HIN)

HIN Instance



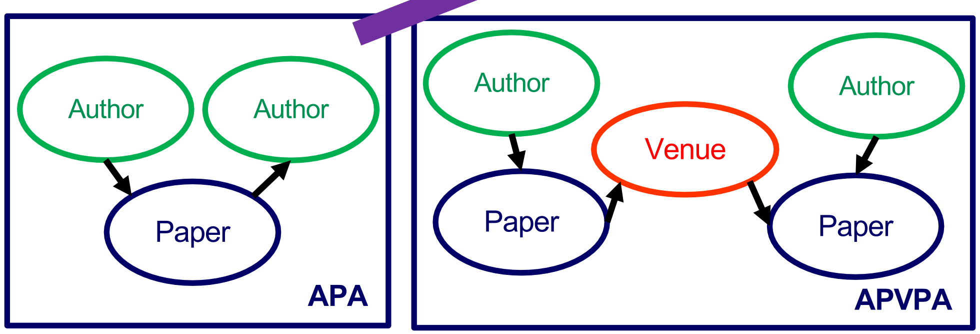
Network Schema
(Meta-level description of HIN)



Path Instance

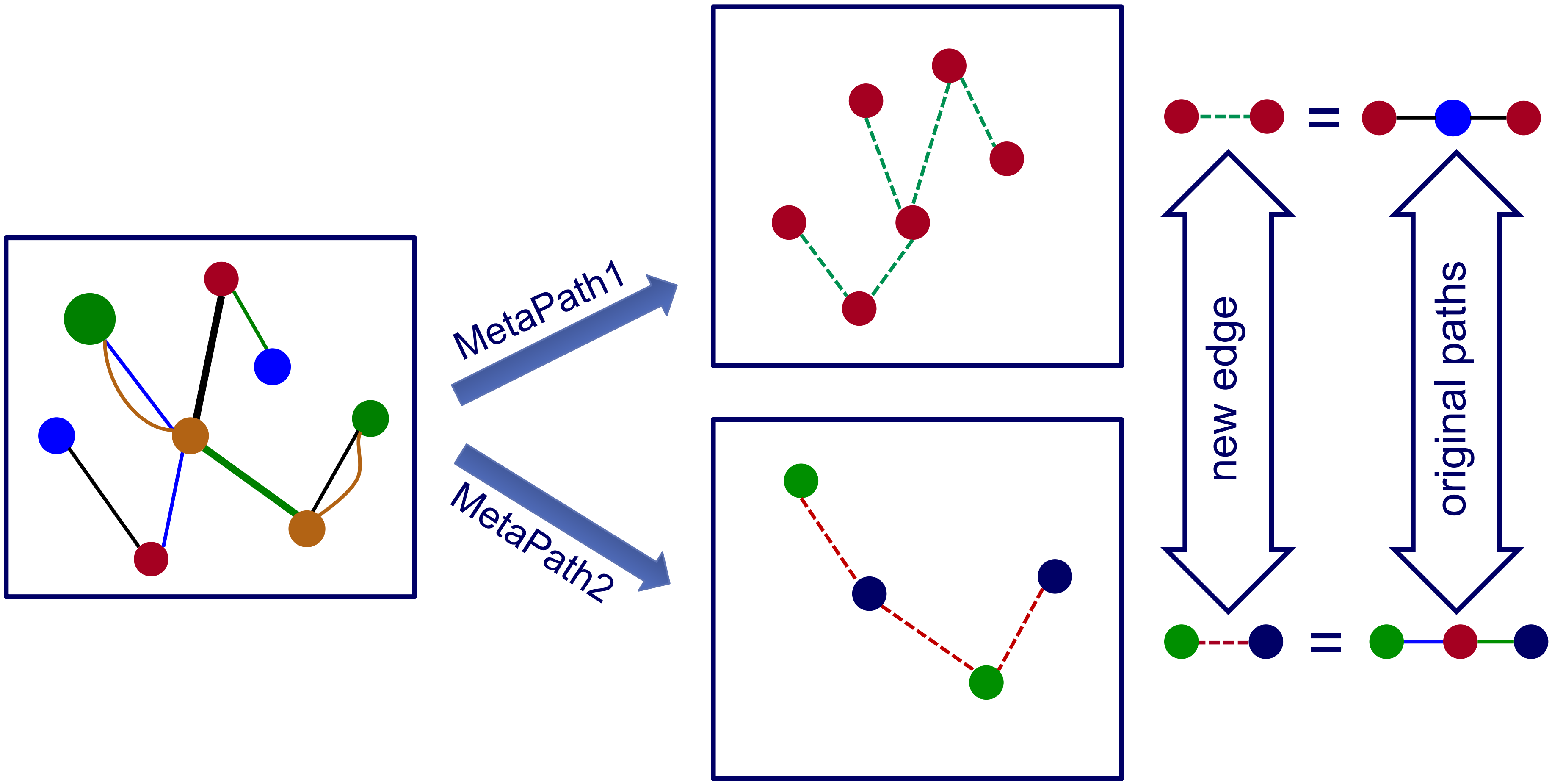
APA
Meaning:
Co-authors
Instances:
Sun-NetClus-Han Sun-PathSim-Yu

MetaPath (Meta-level description of paths)



APVPA
Meaning:
Similar interest colleagues
Instances:
Sun-PathSim-VLDB-PathSim-Han Sun-PathSim-VLDB-GenClus-Aggarwal

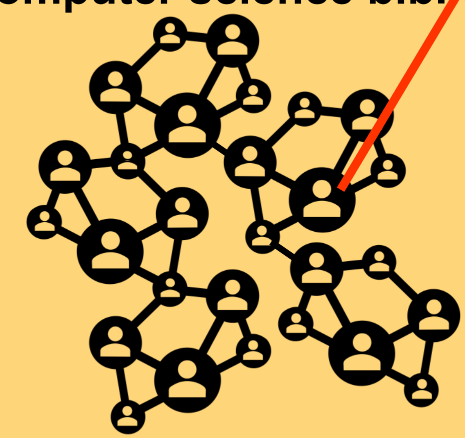
Implicit Meta-path Intuition



Question:

Q: In DBLP who are most similar to "Christos Faloutsos"?

How to define?

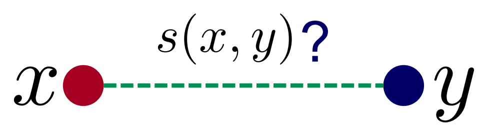


A: PathSim and Meta-path is one way!

Details

PathSim

PathSim: Normalized path count between two nodes x, y following a meta-path P:

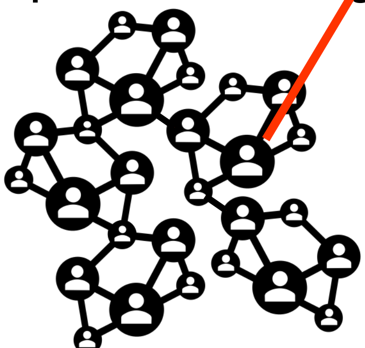


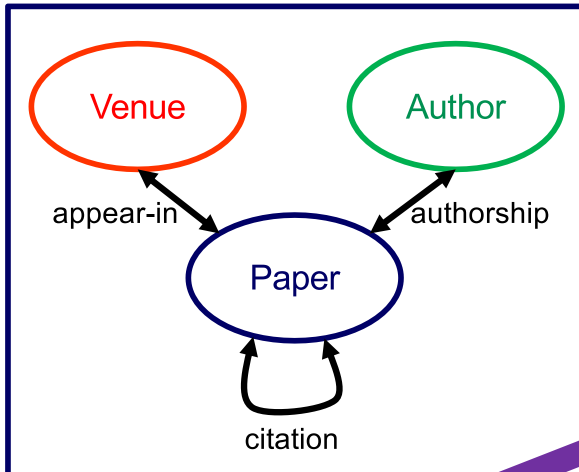
Number of paths between nodes following $\mathcal{P}$

$$s(x,y) = \frac{2 \times |\{p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in \mathcal{P}\}|}{|\{p_{x \rightsquigarrow x} : p_{x \rightsquigarrow x} \in \mathcal{P}\}| + |\{p_{y \rightsquigarrow y} : p_{y \rightsquigarrow y} \in \mathcal{P}\}|}$$

Different Meta-paths Give Different Semantics

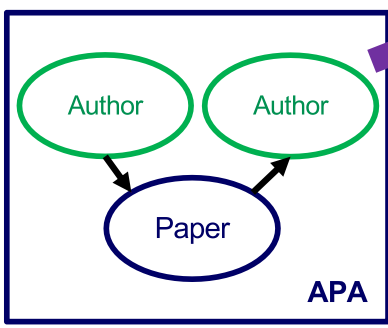
DBLP
computer science bibliography

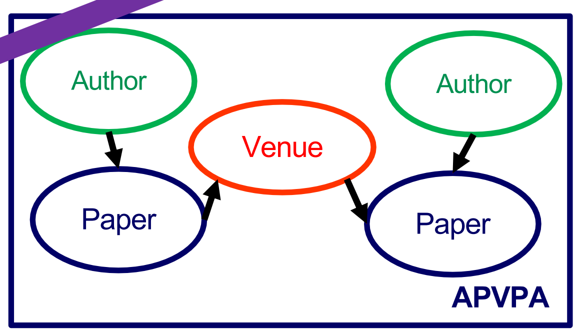




Co-authors

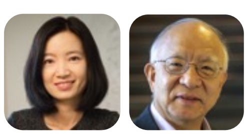
Rank	Author	Score
1	Christos Faloutsos	1
2	Spiros Papadimitriou	0.127
3	Jimeng Sun	0.12
4	Jia-Yu Pan	0.114
5	Agma J. M. Traina	0.110
6	Jure Leskovec	0.096
7	Caetano Traina Jr.	0.096
8	Hanghang Tong	0.091
9	Deepayan Chakrabarti	0.083
10	Flip Korn	0.053





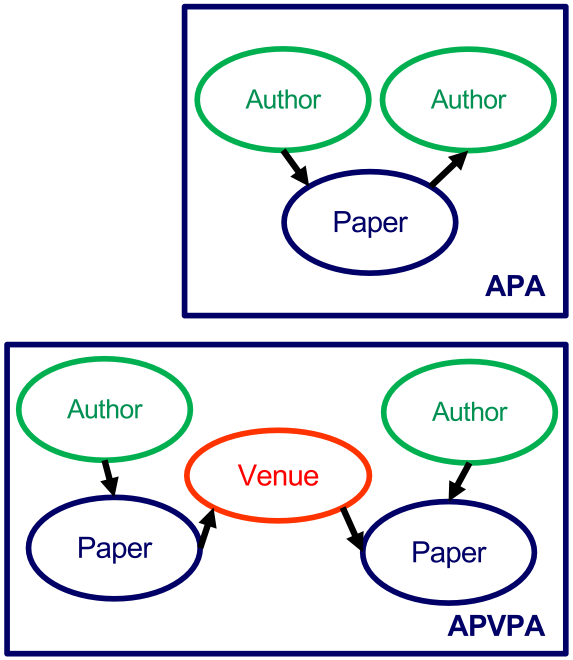
Similar interest colleagues

Rank	Author	Score
1	Christos Faloutsos	1
2	Jiawei Han	0.842
3	Rakesh Agrawal	0.838
4	Jian Pei	0.8
5	Charu C. Aggarwal	0.739
6	H. V. Jagadish	0.705
7	Raghu Ramakrishnan	0.697
8	Nick Koudas	0.689
9	Surajit Chaudhuri	0.677
10	Divesh Srivastava	0.661

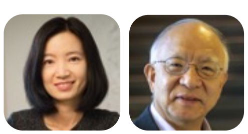


[VLDB] "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks", Sun, Y., Han, J., Yan, X., Yu, P. S., & Wu, T.

Meta-Path



- Similarity and Search: PathSim
- Link Prediction: PathPredict
- Clustering: PathSelClus



[Book] “Mining heterogeneous information networks: principles and methodologies”
Sun, Yizhou, and Jiawei Han

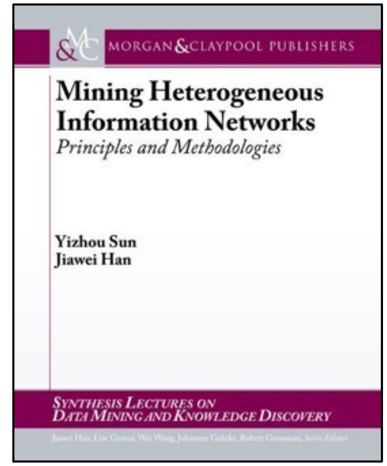
Software Tools

– Hetnetpy: https://het.io/software/#hetnetpy

References

- Shi, C., Li, Y., Zhang, J., Sun, Y., & Philip, S. Y.
  A survey of heterogeneous information network analysis
  IEEE Transactions on Knowledge and Data Engineering, 2016

- Sun, Yizhou, and Jiawei Han.,
  Mining heterogeneous information networks: principles and methodologies
  Synthesis Lectures on Data Mining and Knowledge Discovery, 2012



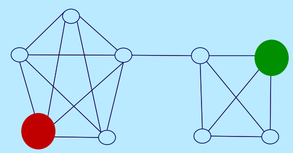
Bird's eye view

Task \ Tool	1.1 PR/HITS	1.1 PPR	1.2 METIS/ SVD	1.3 OddBall+	1.4 BP	2.1 FM	2.1 Tensor	2.2 HIN	2.3 SRL
1.1 Node Ranking	👍					(👍)		(👍)	
1.1' Link Prediction		👍				👍	👍	👍	
1.2 Comm. Detection			👍				👍	(👍)	
1.3 Anomaly Detection				👍			👍		
1.4 Propagation					👍			(👍)	

Part 1: Plain Graphs | **Part 2: Complex Graphs**

Bird's eye view

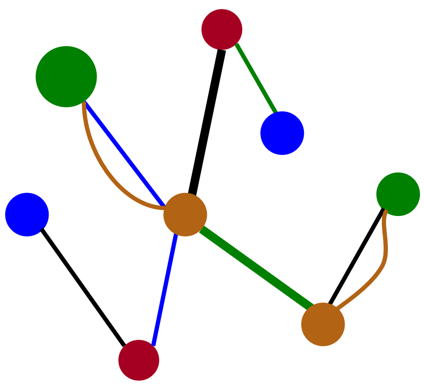
- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
  - P 2.2: Heterogeneous Information Networks
  - P 3.3: Statistical Relational Learning
    - P3.3.1: Node Labeling / Collective Classification
    - P3.3.2: Link Prediction / Recommender Systems
    - P3.3.3: Entity Resolution / Knowledge Graph Identification



amazon

Statistical Relational Learning

Real Data



Dependencies
& Structure

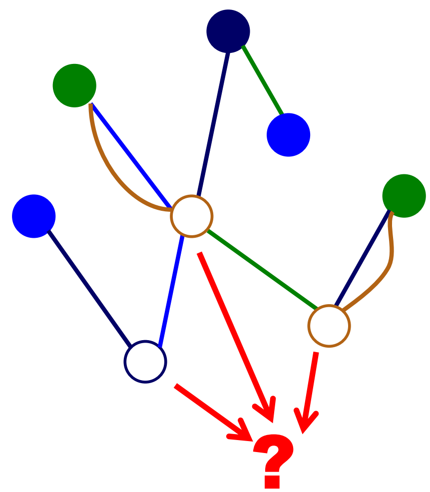
Flattening

Transformed Data

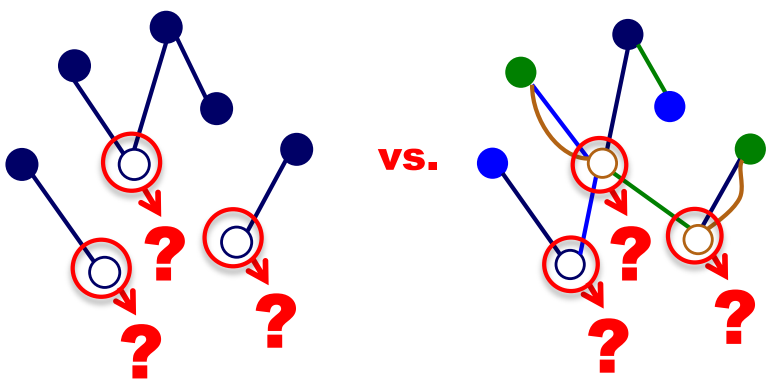
2.26	1.59	1.46	1.23	1.16	1.13	1.10	1.07	1.05	1.03	1.02	1.01
0.84	1.59	1.46	1.23	1.16	1.13	1.10	1.07	1.05	1.03	1.02	1.01
0.71	0.50	1.46	1.23	1.16	1.13	1.10	1..07	1.05	1.03	1.02	1.01
0.42	0.29	0.27	1.23	1.16	1.13	1.10	1.07	1.05	1.03	1.02	1.01
0.32	0.22	0.21	1.17	1.16	1.13	1.10	1.07	1.05	1.03	1.02	1.01
0.26	0.18	0.17	0.14	1.13	1.13	1.10	1.07	1.05	1.03	1.02	1.01
0.20	0.14	0.13	0.11	0.10	0.10	1.10	1.07	1.05	1.03	1.02	1.01
0.15	0.11	0.10	0.08	0.08	0.08	0.07	1.07	1.05	1.03	1.02	1.01
0.11	0.08	0.07	0.06	0.06	0.06	0.05	0.05	1.05	1.03	1.02	1.01
0.07	0.05	0.04	0.04	0.03	0.03	0.03	0.03	0.03	1.03	1.02	1.01
0.05	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02	1.02	1.01
0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	1.01

[Suitable for Most ML Algorithms]

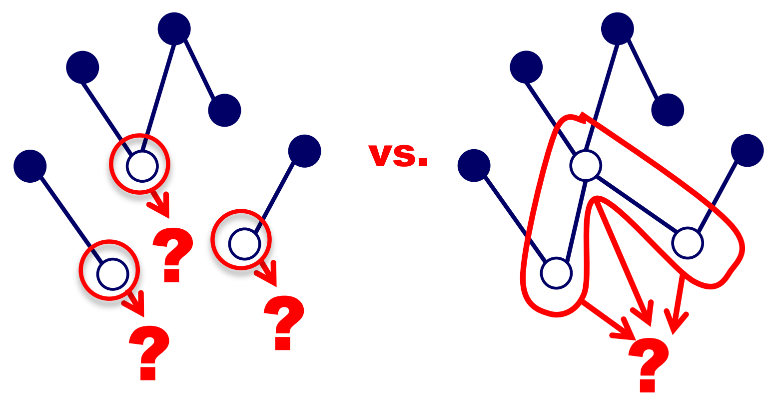
Complex Networks



1. Capture multi-relational nature

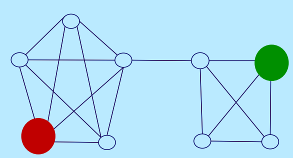


Joint Inference



Bird's eye view

- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
  - P 2.2: Heterogeneous Information Networks
  - P 3.3: Statistical Relational Learning
    - P3.3.1: Node Labeling / Collective Classification
    - P3.3.2: Link Prediction / Recommender Systems
    - P3.3.3: Entity Resolution / Knowledge Graph Identification

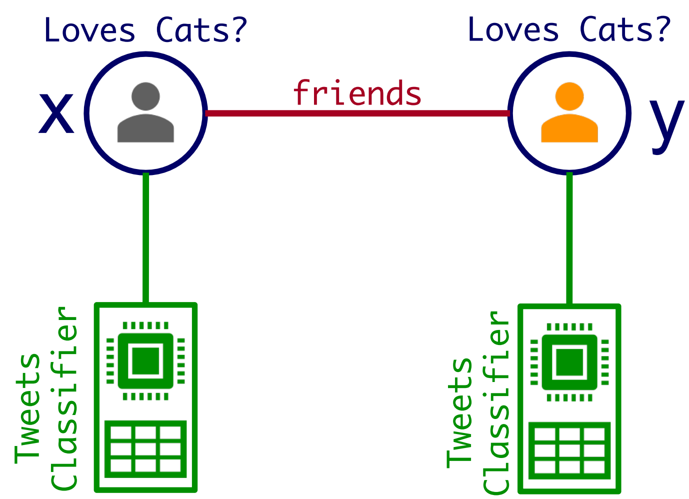


Question:

Q: How can we propagate labels in networks with extra information?

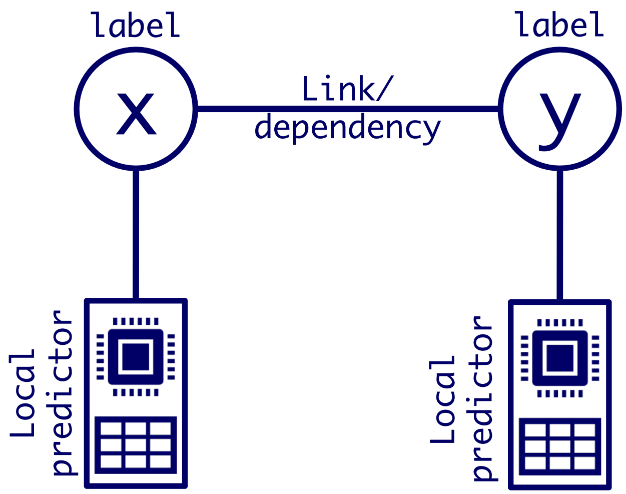
A: Statistical Relational Learning is one way.

SRL: Node Labeling [Collective Classification]

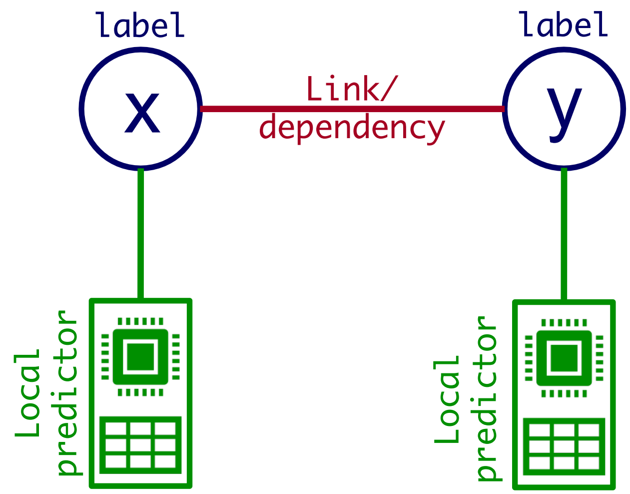


How can we use the friendship relation to improve the predictions?

SRL: Node Labeling [Collective Classification]



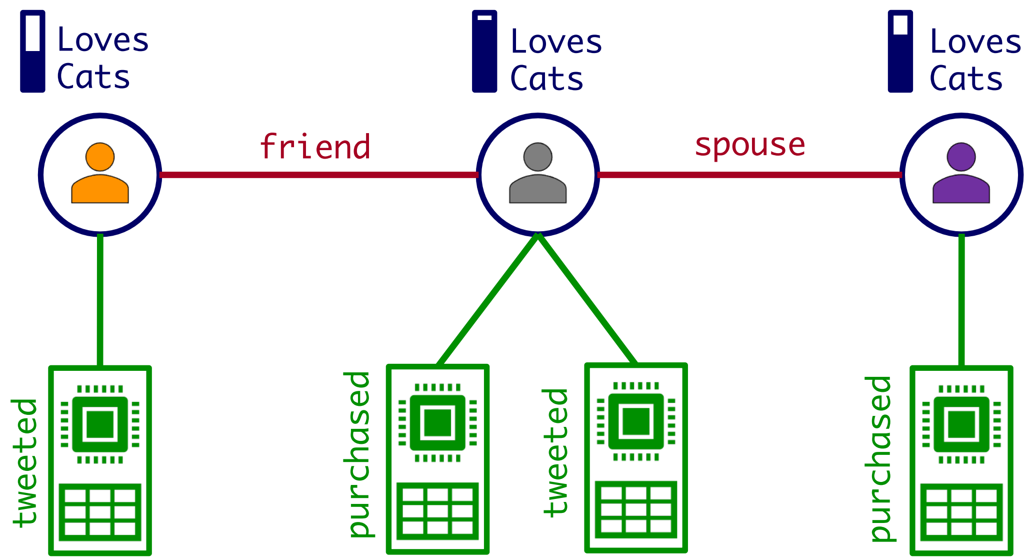
SRL: Node Labeling [Collective Classification]



SRL Answer:

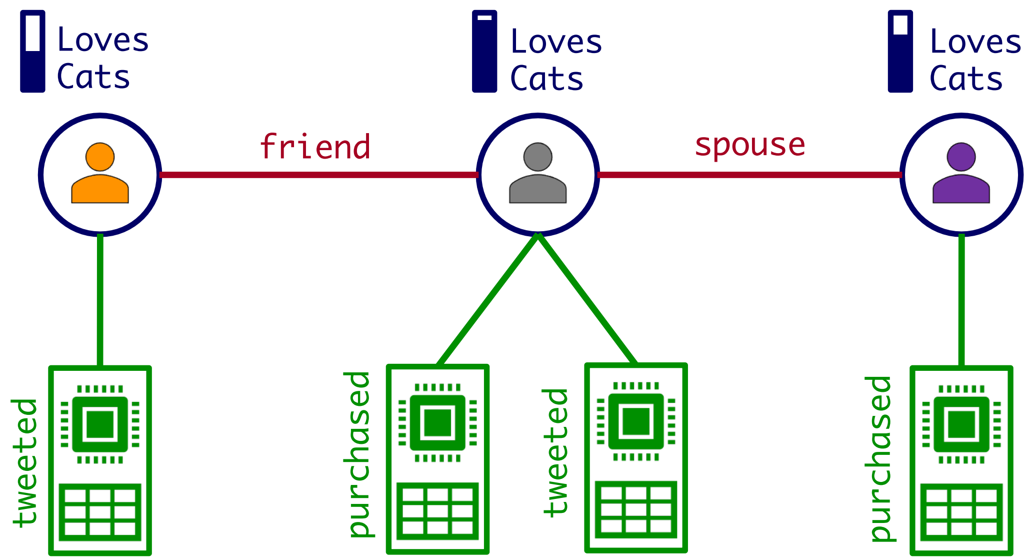
```
    local-predictor(x,l) -> label(x,l)
  label(x,l) & link(x,y) -> label(y,l)
```

SRL: Node Labeling [Collective Classification]




```
purchased(x, "cat_food") -> loves(x, "Cats")
tweeted(x, "#catslover") -> loves(x, "Cats")

loves(x, "Cats") & spouse(x,y) -> loves(y, "Cats")
loves(x, "Cats") & friend(x,y) -> loves(y, "Cats")
```

SRL: Node Labeling [Collective Classification]

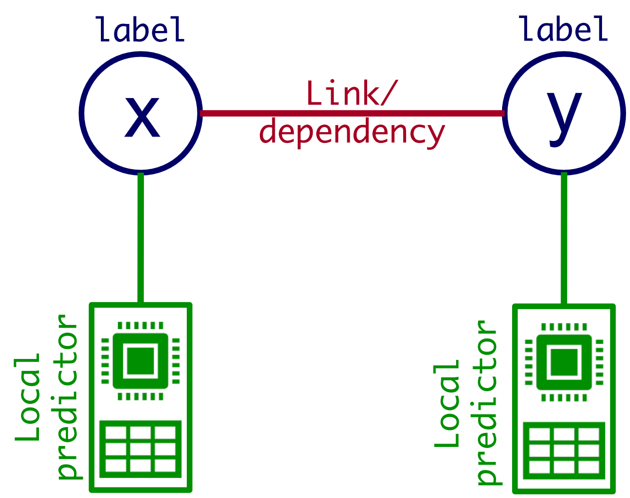


Weights:
- Given by user or
- Learned from data

Soft-values

```
0.9: purchased(x, "cat_food") -> loves(x, "Cats")
0.8: tweeted(x, "#catslover") -> loves(x, "Cats")

0.85: loves(x, "Cats") & spouse(x,y) -> loves(y, "Cats")
0.7:  loves(x, "Cats") & friend(x,y) -> loves(y, "Cats")
```

SRL: Node Labeling [Collective Classification]

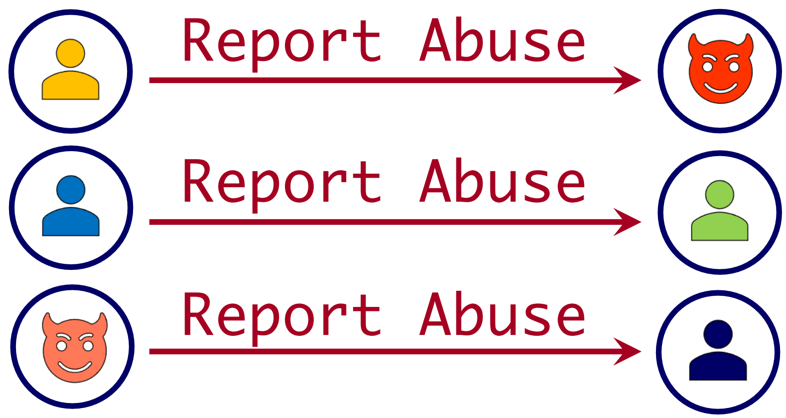


SRL Answer:

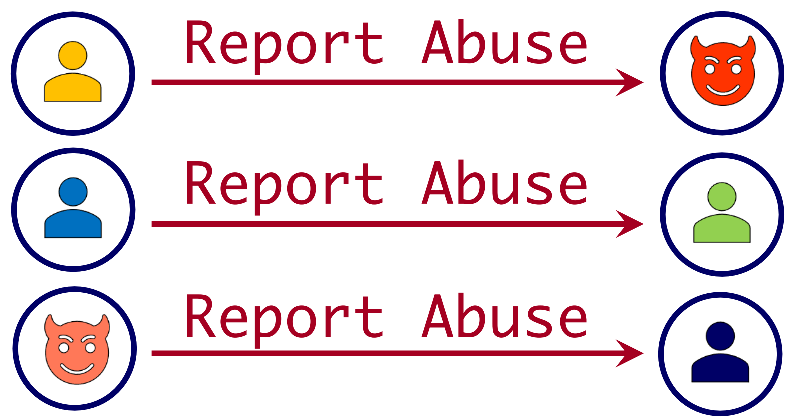
```
    local-predictor(x,l) -> label(x,l)
  label(x,l) & link(x,y) -> label(y,l)
```

Social Spammer Detection

Social Spammer Detection

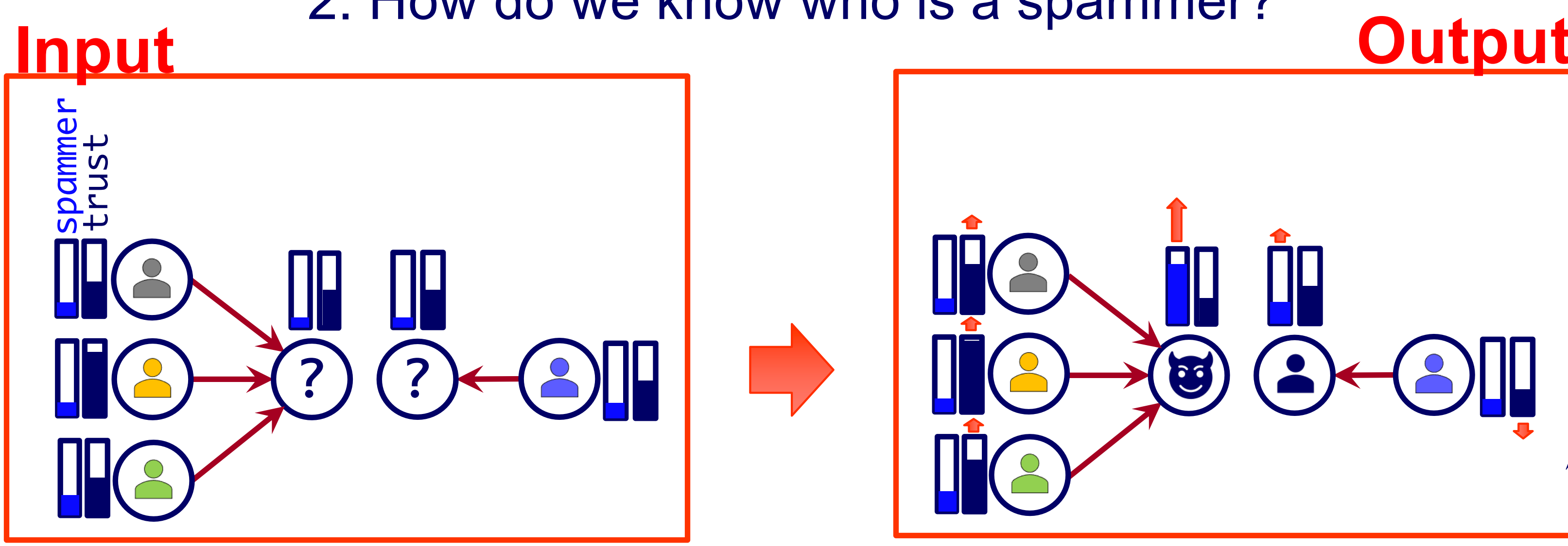


Social Spammer Detection



Task:

1. How do we know who is telling the truth?
2. How do we know who is a spammer?

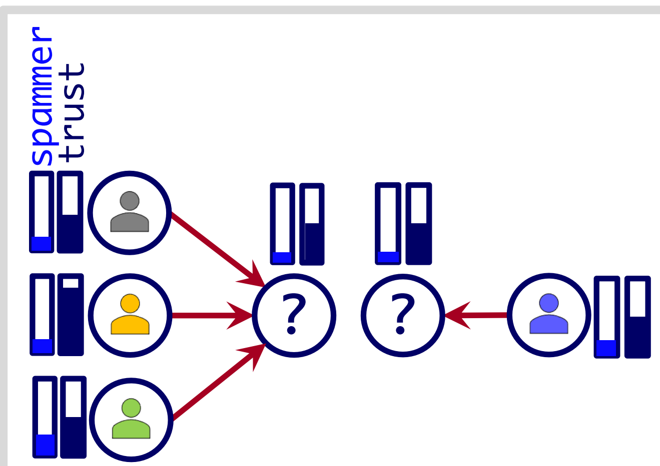


SRL Answer

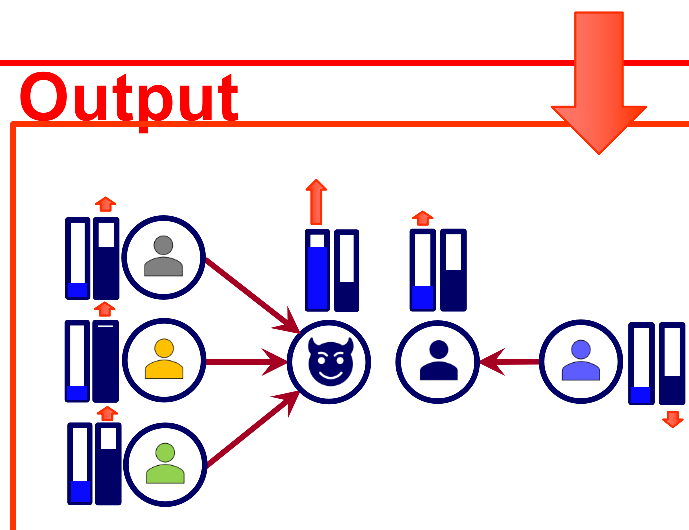
Input

```
              prior-trust(x) ->  trusted(x)

     trusted(x) & reported(x,y) ->  spammer(y)

     spammer(y) & reported(x,y) ->  trusted(x)
    ~spammer(y) & reported(x,y) -> ~trusted(x)

                                   ~spammer(x)
```

+



Output



Statistical Relational Learning Frameworks

Alchemy (MLN)  https://alchemy.cs.washington.edu/

Probabilistic Soft Logic  https://psl.linqs.org/

Felix (Tuffy)  http://i.stanford.edu/hazy/felix/

How using PSL looks like:

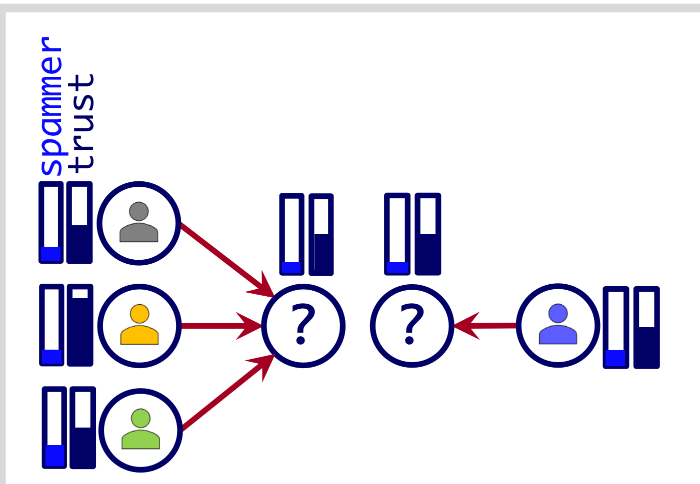
Input

Templates

```
              prior-trust(x) ->  trusted(x)

  trusted(x) & reported(x,y) ->  spammer(y)

  spammer(y) & reported(x,y) ->  trusted(x)
 ~spammer(y) & reported(x,y) -> ~trusted(x)

                                ~spammer(x)
```

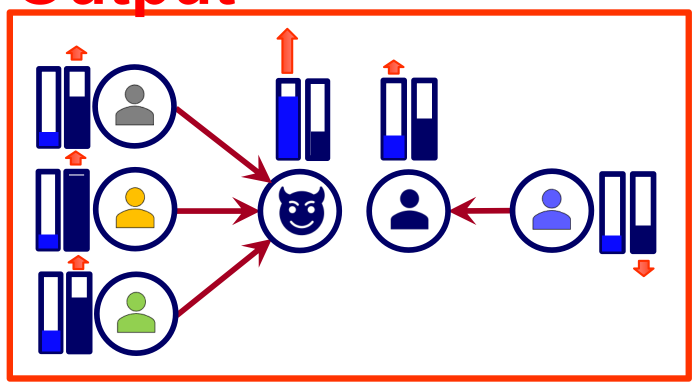
+

Data



PSL

Output



How using PSL looks like:

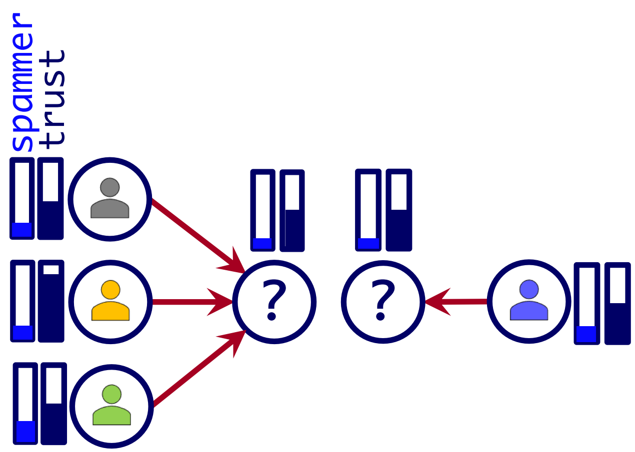
Input

Templates

```
               prior-trust(x) ->  trusted(x)

 trusted(x) & reported(x,y) ->  spammer(y)

 spammer(y) & reported(x,y) ->  trusted(x)
~spammer(y) & reported(x,y) -> ~trusted(x)

                               ~spammer(x)
```

+

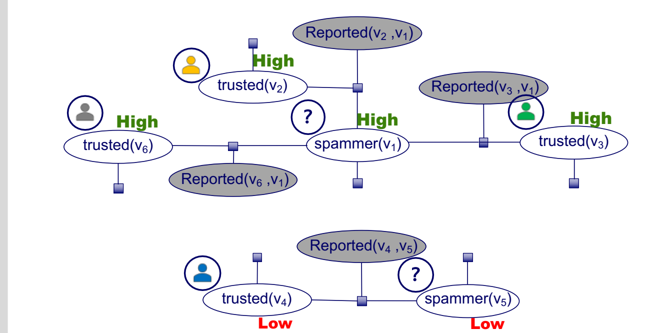
Data



PSL

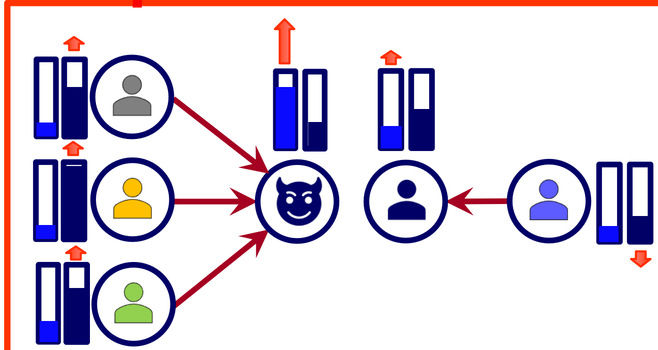
Grounding & Inference

(under the hood)

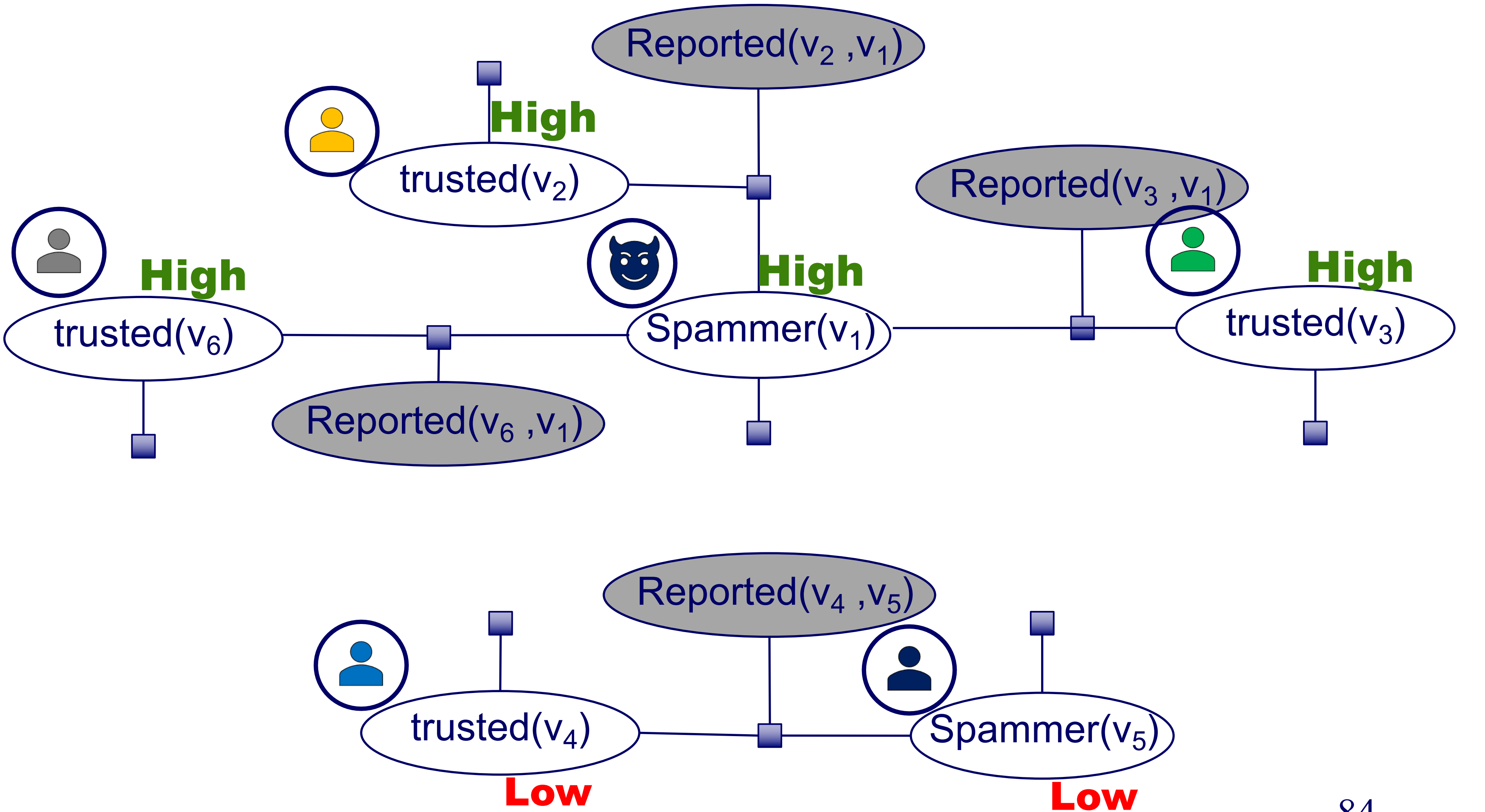


$$P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z(\lambda)} \exp\left[-\sum_{j=1}^{m} \lambda_j \phi_j(\mathbf{Y},\mathbf{X})\right]$$

Output

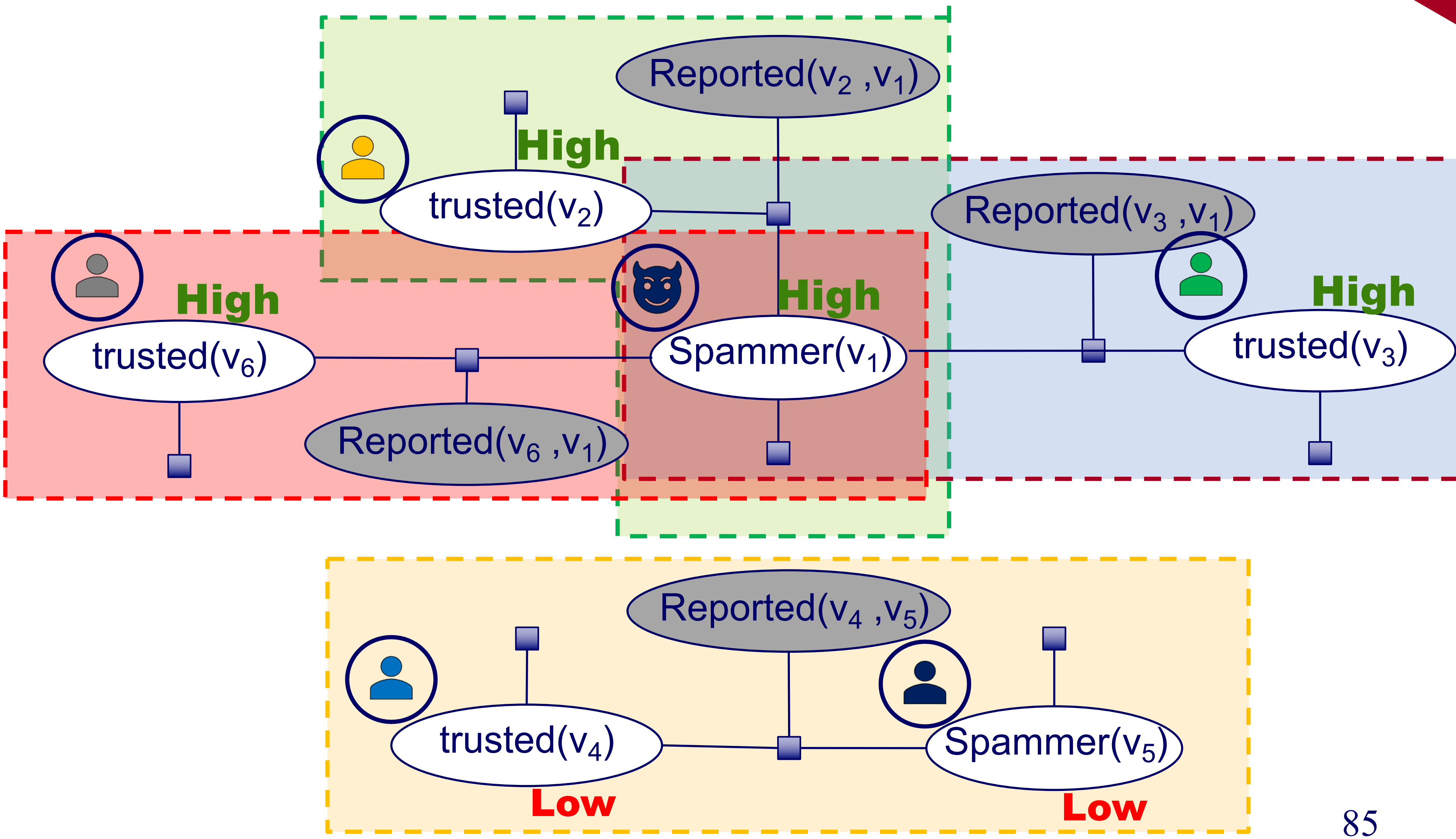


Desired PGM Model



Templates

Under the hood!



Under the hood!

Probabilistic Soft Logic (PSL): A templating language with first-order logic syntax

Example PSL rule:

$$\texttt{w: } \overbrace{\texttt{trusted(x) \& reported(x,y)}}^{r_{body}} \texttt{ -> } \overbrace{\texttt{spammer(y)}}^{r_{head}}$$

Under the hood!

Probabilistic Soft Logic (PSL):

$$\underbrace{\text{A templating language with first-order logic syntax}}$$

Example PSL rule:

$$\texttt{w:}\ \overbrace{\underbrace{\texttt{trusted(x)}}_{0\ \le\ \text{Soft truth}\ \le\ 1}\ \texttt{\&}\ \texttt{reported(x,y)}}^{r_{body}}\ \texttt{->}\ \overbrace{\texttt{spammer(y)}}^{r_{head}}$$

$$\begin{cases} p\ \tilde{\wedge}\ q = \max(0,\ p + q - 1) \\ p\ \tilde{\vee}\ q = \min(1,\ p + q) \\ \quad \neg p = 1 - p \end{cases}$$

amazon

Under the hood!

Probabilistic Soft Logic (PSL): A templating language with first-order logic syntax

Example PSL rule:

r_{body} r_{head}

```
w: trusted(x) & reported(x,y) -> spammer(y)
```

0 ≤ Soft truth ≤ 1

$$\begin{cases} p \,\tilde{\wedge}\, q = \max(0,\ p+q-1) \\ p \,\tilde{\vee}\, q = \min(1,\ p+q) \\ \neg p = 1 - p \end{cases}$$

Ground

```
w: trusted("alice") & reported("alice","bob") -> spammer("bob")
```

0.7 1 ?

Under the hood!

Probabilistic Soft Logic (PSL): A templating language with first-order logic syntax

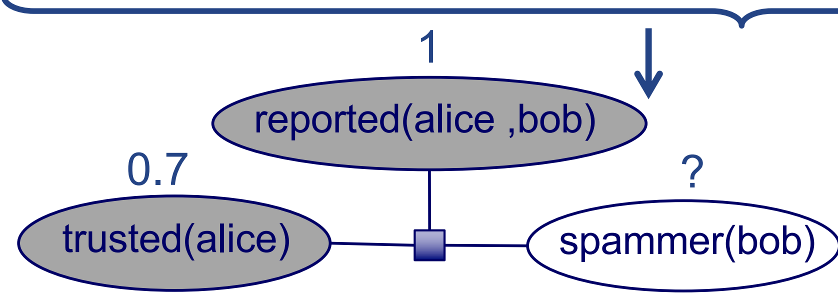
Example PSL rule:

$$\texttt{w: } \underbrace{\overbrace{\texttt{trusted(x)}}_{0 \le \text{Soft truth} \le 1} \texttt{ \& reported(x,y)}}^{r_{body}} \texttt{ -> } \overbrace{\texttt{spammer(y)}}^{r_{head}}$$

$$\begin{cases} p \tilde{\wedge} q = \max(0,\ p + q - 1) \\ p \tilde{\vee} q = \min(1,\ p + q) \\ \neg p = 1 - p \end{cases}$$

Ground

$$\texttt{w: } \underbrace{\overset{0.7}{\texttt{trusted("alice")}} \texttt{ \& } \overset{1}{\texttt{reported("alice","bob")}} \texttt{ -> } \overset{?}{\texttt{spammer("bob")}}}$$



Under the hood!

Probabilistic Soft Logic (PSL): A templating language with first-order logic syntax

Example PSL rule:

```
w: trusted(x) & reported(x,y) -> spammer(y)
```

r_{body} = trusted(x) & reported(x,y)

r_{head} = spammer(y)

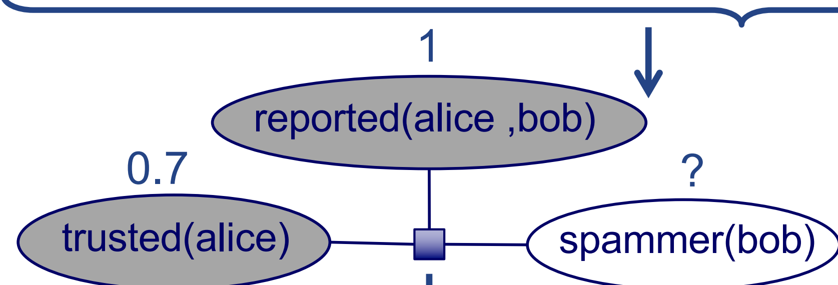
0 ≤ Soft truth ≤ 1

$$\begin{cases} p \,\tilde{\wedge}\, q = \max(0,\ p+q-1) \\ p \,\tilde{\vee}\, q = \min(1,\ p+q) \\ \neg p = 1-p \end{cases}$$

Ground

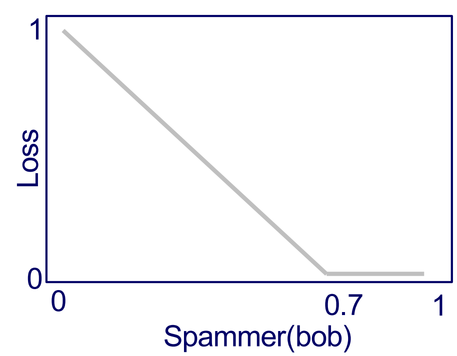
```
w: trusted("alice") & reported("alice","bob") -> spammer("bob")
```

0.7 1 ?



distance to satisfaction

$$\ell = \max\{\mathrm{value}(r_{\mathrm{body}}) - \mathrm{value}(r_{\mathrm{head}}), 0\}$$



Under the hood!

Probabilistic Soft Logic (PSL): A templating language with first-order logic syntax

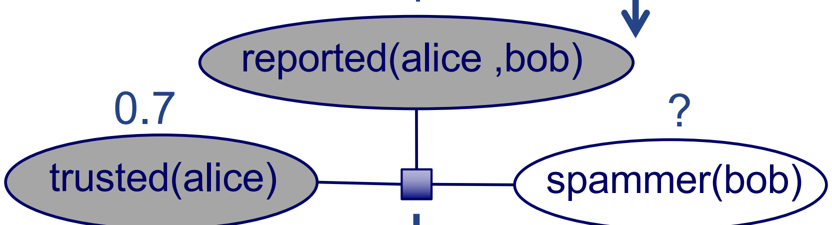
Example PSL rule:

$$\texttt{w: }\underbrace{\underbrace{\texttt{trusted(x)}}_{0 \le \text{Soft truth} \le 1}\texttt{ \& reported(x,y)}}_{r_{body}}\texttt{ -> }\underbrace{\texttt{spammer(y)}}_{r_{head}}$$

$$\begin{cases} p \tilde{\wedge} q = \max(0,\ p+q-1) \\ p \tilde{\vee} q = \min(1,\ p+q) \\ \neg p = 1-p \end{cases}$$

Ground

$$\texttt{w: }\underbrace{\texttt{trusted(“alice”)}}_{0.7}\texttt{ \& }\underbrace{\texttt{reported(“alice”,”bob”)}}_{1}\texttt{ -> }\underbrace{\texttt{spammer(“bob”)}}_{?}$$



distance to satisfaction

$$\ell = \max\{\text{value}(r_{\text{body}}) - \text{value}(r_{\text{head}}), 0\} \longrightarrow \phi_j(\mathbf{Y}, \mathbf{X}) = [\ell_j(\mathbf{Y}, \mathbf{X})]$$

$$P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z(\lambda)} \exp\left[-\sum_{j=1}^{m} \lambda_j \underbrace{\phi_j(\mathbf{Y}, \mathbf{X})}\right]$$

Unknown variables: $\mathbf{Y}$

Evidence: $\mathbf{X}$

How using PSL looks like:

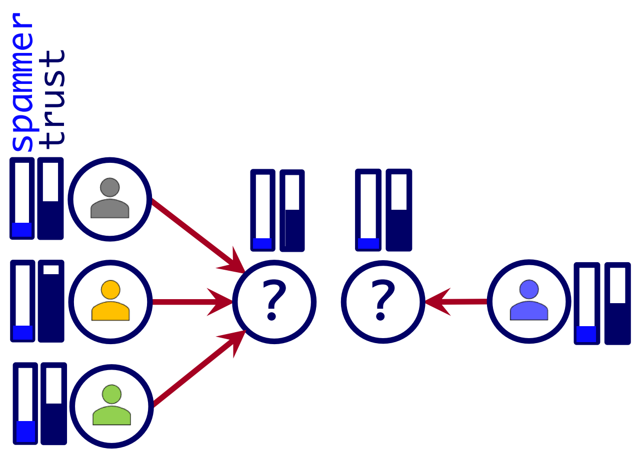
Input

Templates

```
            prior-trust(x) ->  trusted(x)

 trusted(x) & reported(x,y) ->  spammer(y)

 spammer(y) & reported(x,y) ->  trusted(x)
~spammer(y) & reported(x,y) -> ~trusted(x)

                               ~spammer(x)
```

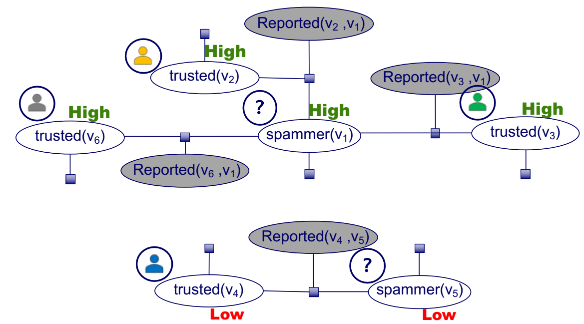
+

Data



PSL

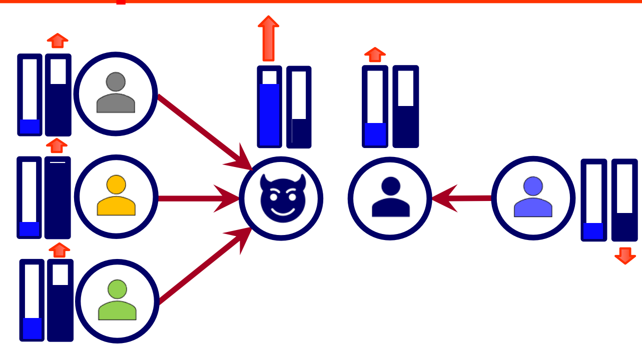
Grounding & Inference



$$P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z(\lambda)} \exp\left[-\sum_{j=1}^{m} \lambda_j \phi_j(\mathbf{Y}, \mathbf{X})\right]$$

(under the hood)

Output



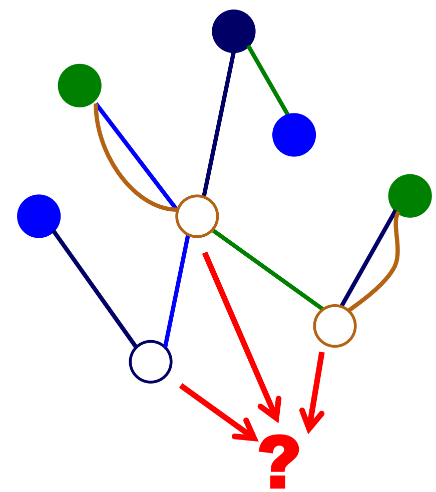
Social Spammer Detection

- Collective model:

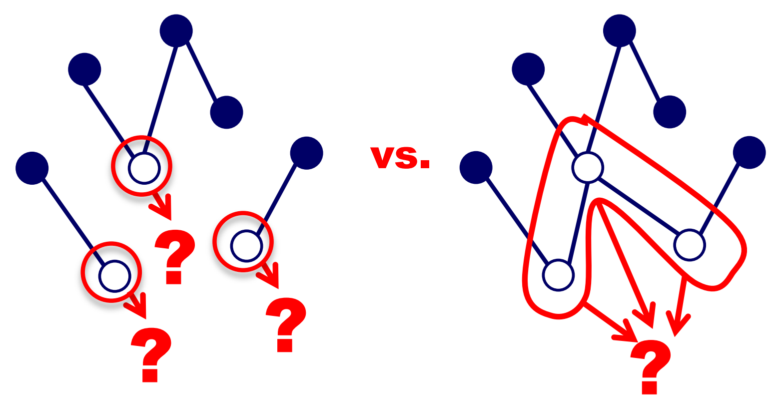
```
                  prior-trust(x) ->  trusted(x)

 trusted(x) & reported(x,y) ->  spammer(y)

 spammer(y) & reported(x,y) ->  trusted(x)
~spammer(y) & reported(x,y) -> ~trusted(x)

                                ~spammer(x)
```

Experiments	AU-PR	AU-ROC
Collective Classification	**0.790** ± 0.005	**0.788** ± 0.003

Node Classification



Joint Inference



Report Model

- Collective model:

```
               prior-trust(x) ->  trusted(x)

  trusted(x) & reported(x,y) ->  spammer(y)

  spammer(y) & reported(x,y) ->  trusted(x)
 ~spammer(y) & reported(x,y) -> ~trusted(x)

                                ~spammer(x)
```

- Non-collective model ($\simeq$ weighted sum of the reports):

```
prior-trust(x) & reported(x,y) ->  spammer(y)

                                  ~spammer(x)
```

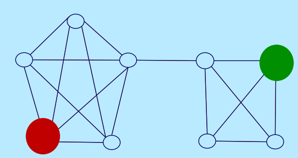
Classification Using Reports

Experiments	AU-PR	AU-ROC
Non-collective model	0.690 ± 0.003	0.624 ± 0.001
Collective model	**0.790 ± 0.005**	**0.788 ± 0.003**

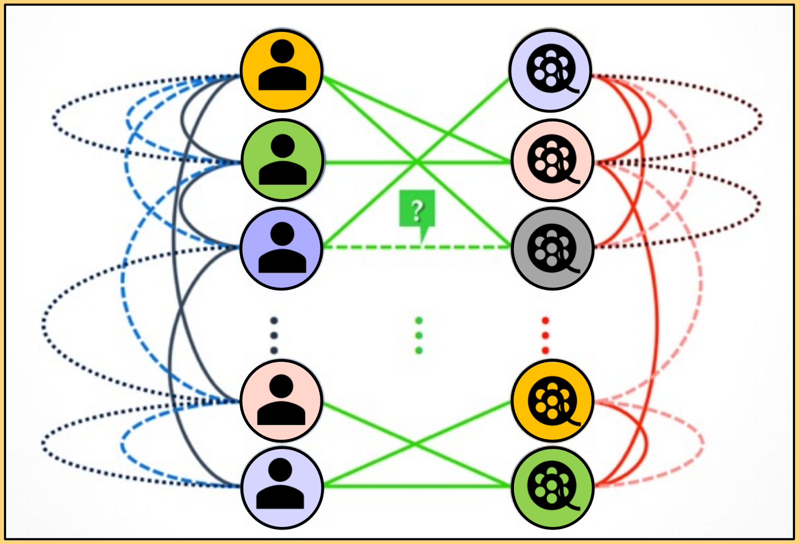
[KDD] "Collective Spammer Detection in Evolving Multi-Relational Social Networks", Fakhraei, S., Foulds, J., Shashanka, M., & Getoor, L.

Bird's eye view

- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
  - P 2.2: Heterogeneous Information Networks
  - P 3.3: Statistical Relational Learning
    - P3.3.1: Node Labeling / Collective Classification
    - P3.3.2: Link Prediction / Recommender Systems
    - P3.3.3: Entity Resolution / Knowledge Graph Identification

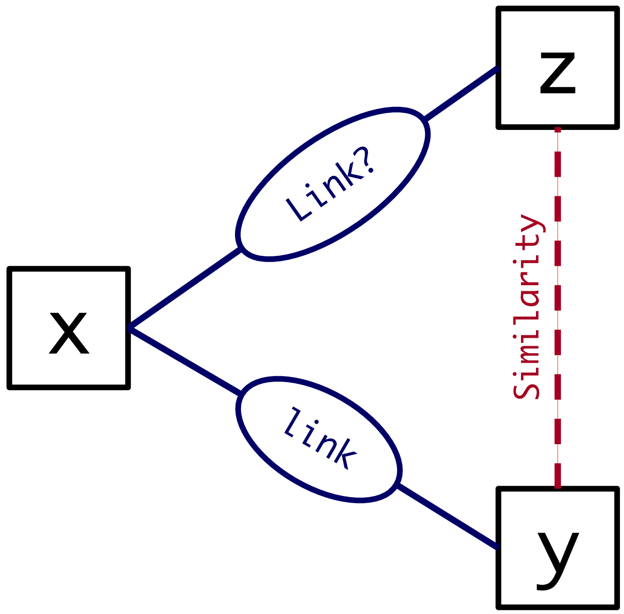


Question:

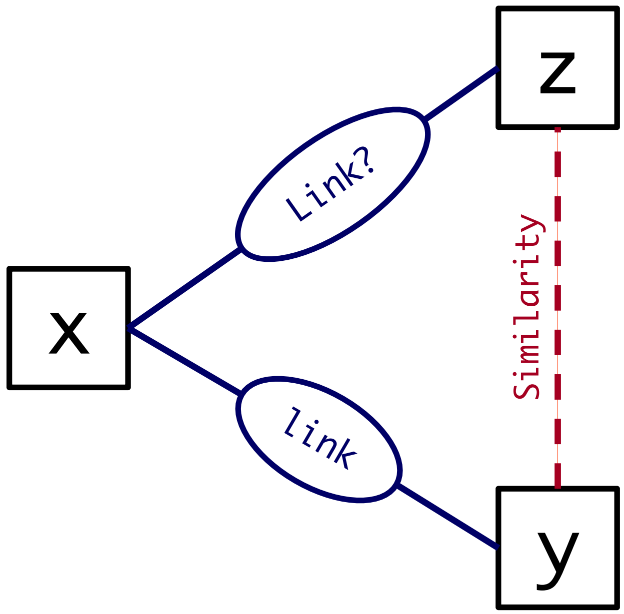


1. How can we use multiple similarities between nodes to infer link values?
2. How can we propagate link information?
3. How can we add additional model signals?

Link Inference Pattern

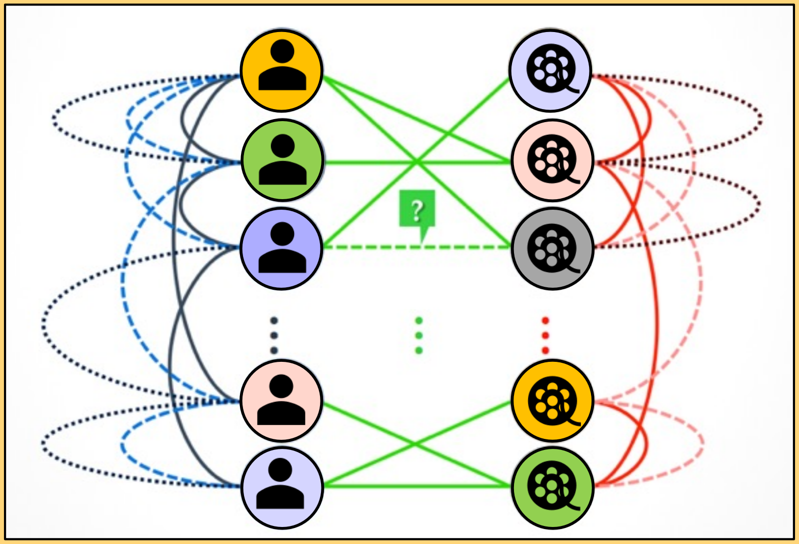


Link Inference Template




```
w: link(x,y) & similar(y,z) -> link(x,z)
```

Question:



? 1. How can we use multiple similarities between nodes to infer link values?

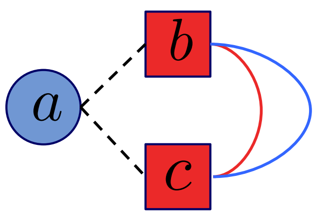
? 2. How can we propagate link information?

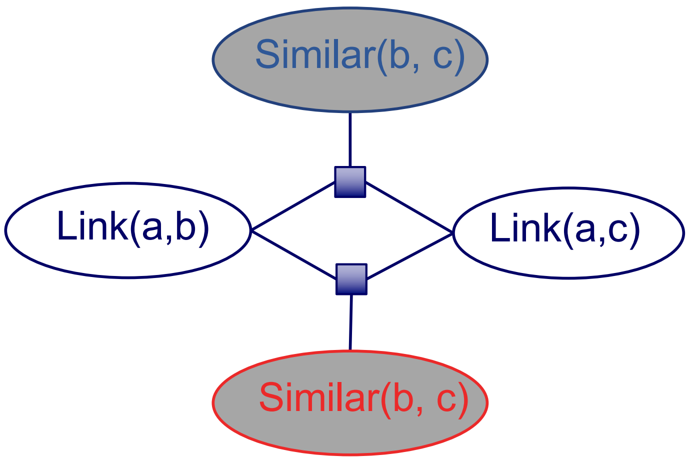
3. How can we add additional model signals?

Link Inference Model Characteristics

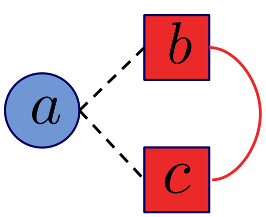
Details

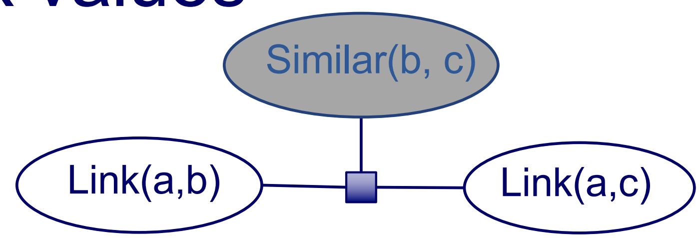
- Support multiple relations



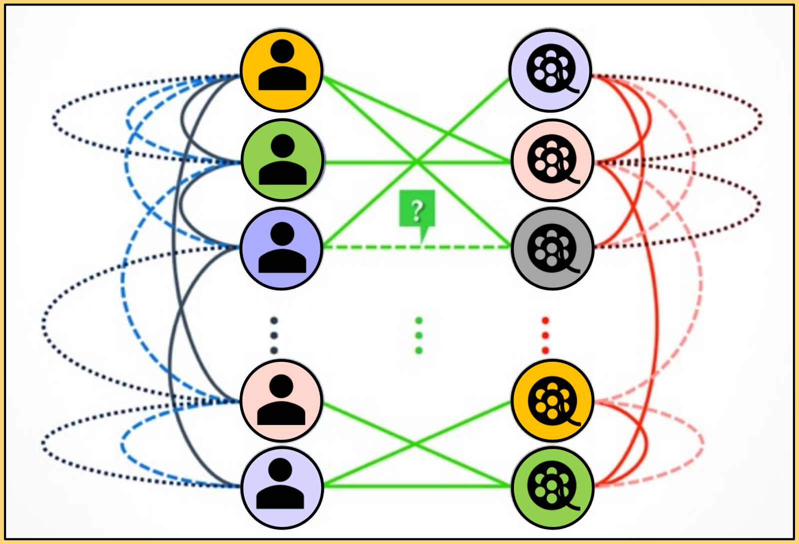


- Joint inference of link values



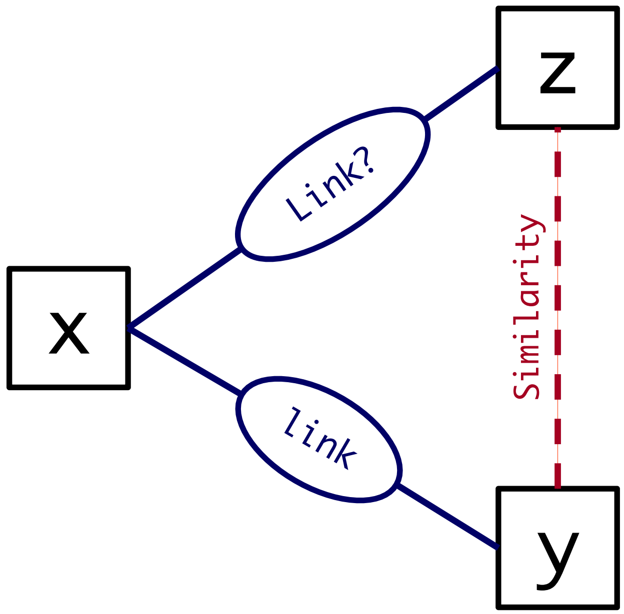


Question:



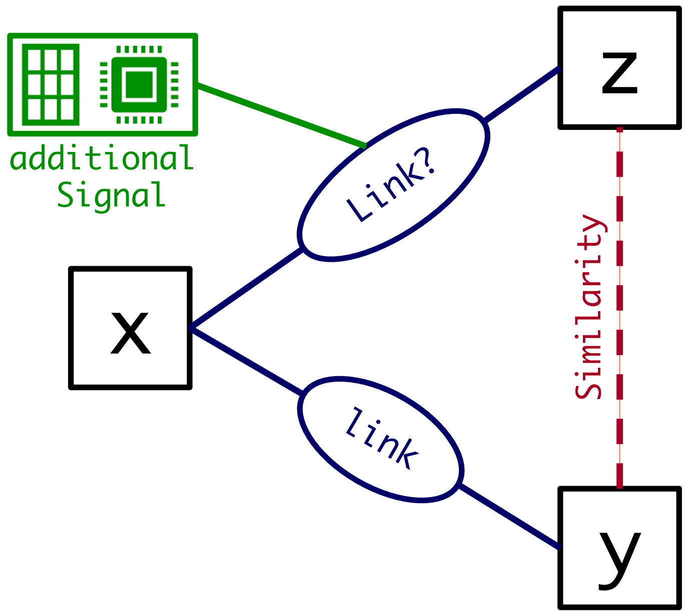
1. How can we use multiple similarities between nodes to infer link values?
2. How can we propagate link information?
3. How can we add additional model signals?

Can we add additional signals?




```
link(x,y) & similar(y,z) -> link(x,z)
```

Can we add additional signals?

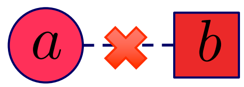



```
link(x,y) & similar(y,z) -> link(x,z)
  additional-signal(x,y) -> link(x,y)
```

Details

Typical Additional Signals

- Enforce sparsity

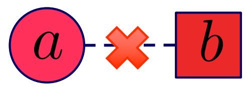


~rating(u,i)

Details

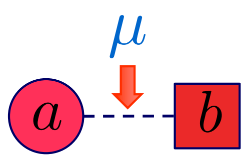
Typical Additional Signals

- Enforce sparsity




```
~rating(u,i)
```

- Distribution statistics

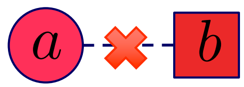



```
mean-rating-user(u) -> rating(u,i)
```

Details

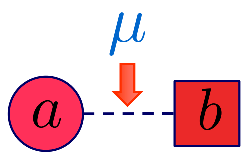
Typical Additional Signals

– Enforce sparsity



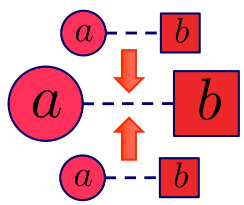
`~rating(u,i)`

– Distribution statistics



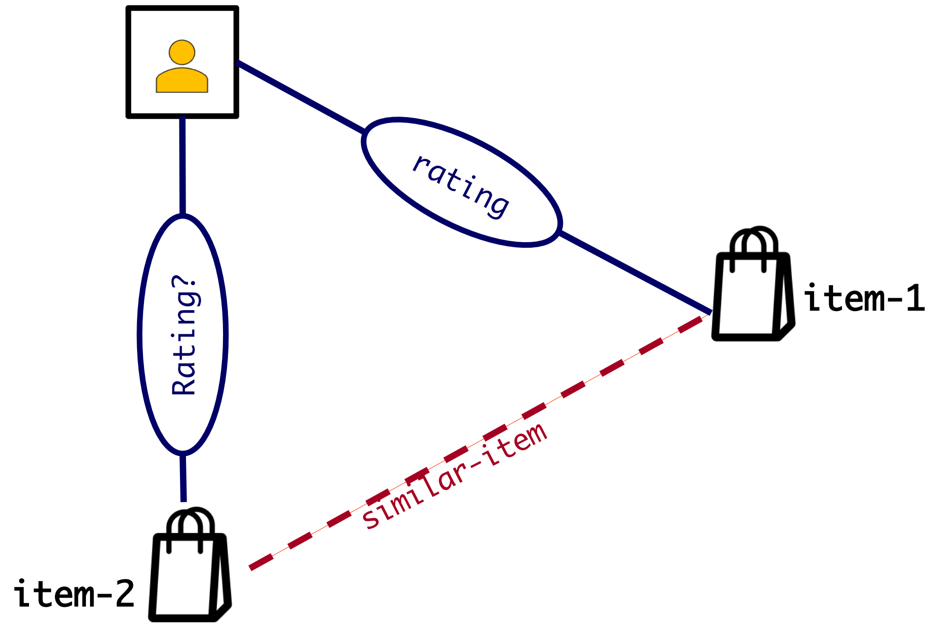
`mean-rating-user(u) -> rating(u,i)`

– Predictions from other models



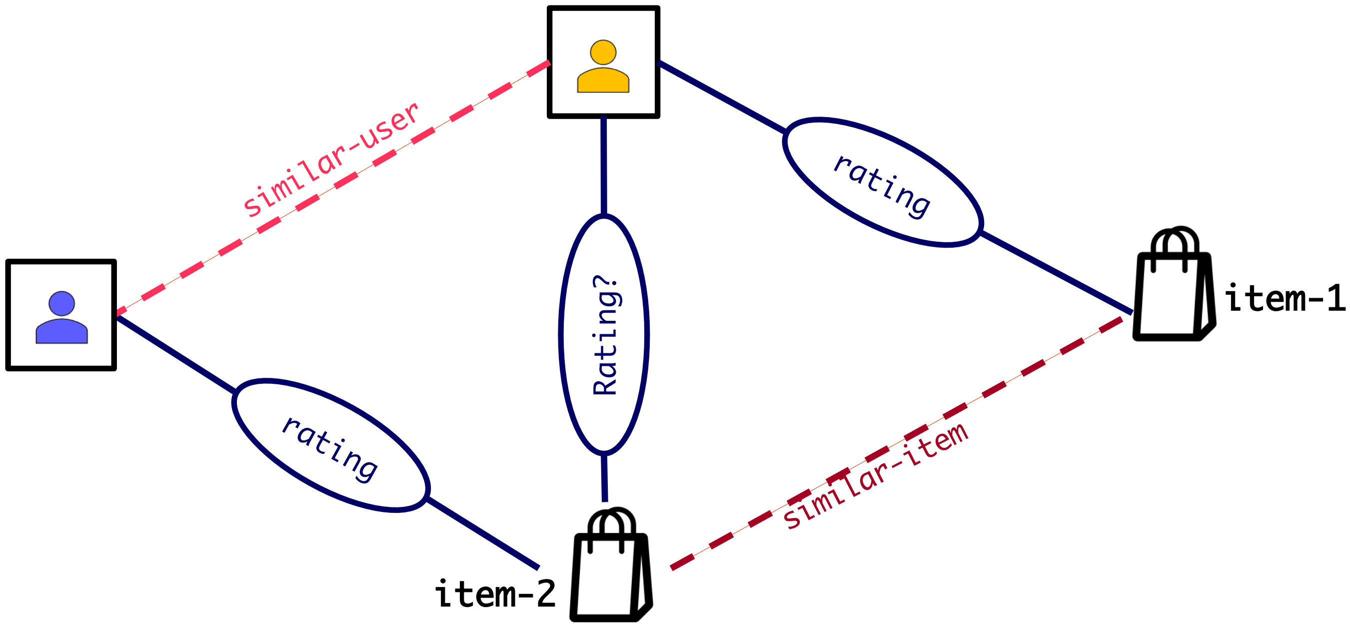
`FM-rating(u,i) -> rating(u,i)`

Template for Recommender Systems



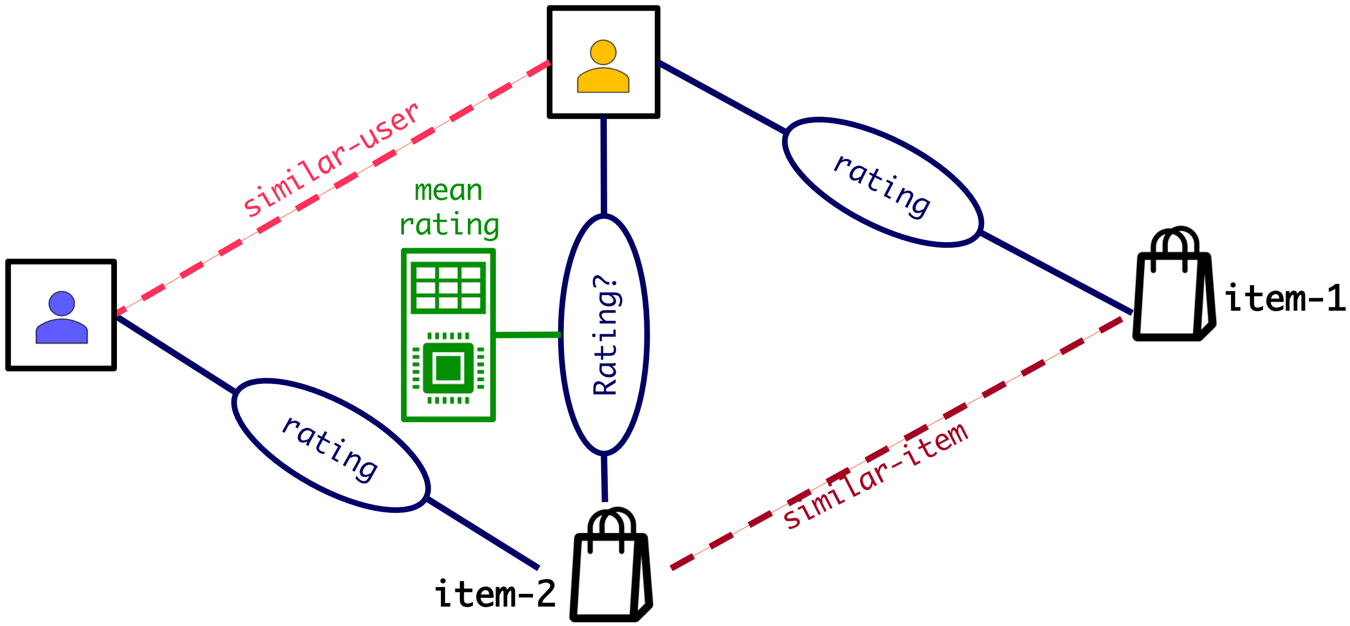
rating(u,i1) & similar-item(i1,i2) -> rating(u,i2)

Template for Recommender Systems



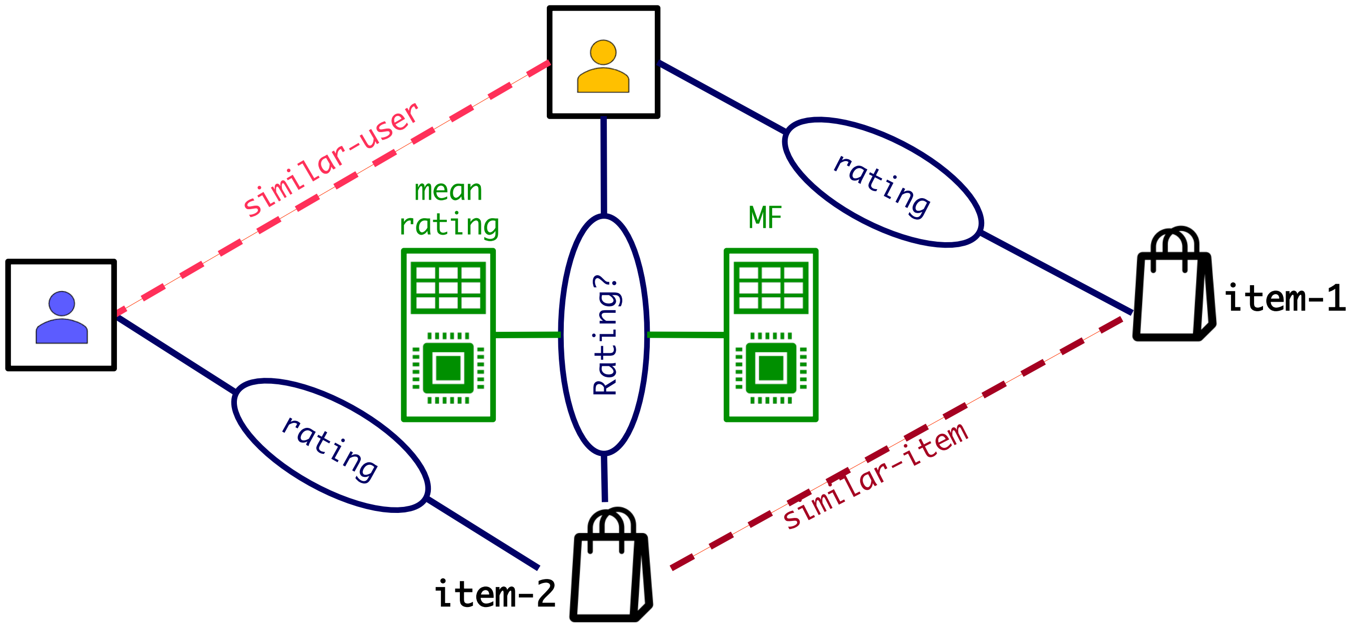

```
rating(u,i1) & similar-item(i1,i2) -> rating(u,i2)
rating(u1,i) & similar-user(u1,u2) -> rating(u2,i)
```

Template for Recommender Systems




```
rating(u,i1) & similar-item(i1,i2) -> rating(u,i2)
rating(u1,i) & similar-user(u1,u2) -> rating(u2,i)

                 mean-rating-user(u) -> rating(u,i)
```

Template for Recommender Systems




```
rating(u,i1) & similar-item(i1,i2) -> rating(u,i2)
rating(u1,i) & similar-user(u1,u2) -> rating(u2,i)

          mean-rating-user(u) -> rating(u,i)
          mean-rating-item(i) -> rating(u,i)

          MF-rating(u,i) -> rating(u,i)
```

Experimental Validation

Dataset	Yelp	Last.fm
No. of users	34,454	1,892
No. of items	3,605	17,632
No. of ratings	99,049	92,834
Content	514 business categories	9,719 artist tags
Social	81,512 friendships	12,717 friendships
Sparsity	99.92%	99.72%

		Yelp		Last.fm	
	Model	RMSE (SD)	MAE (SD)	RMSE (SD)	MAE (SD)
Base models	Item-based	1.216 (0.004)	0.932 (0.001)	1.408 (0.010)	1.096 (0.008)
	MF	1.251 (0.006)	0.944 (0.005)	1.178 (0.003)	0.939 (0.003)
	BPMF	1.191 (0.003)	0.954 (0.003)	1.008 (0.005)	0.839 (0.004)
Hybrid models	Naive hybrid (averaged predictions)	1.179 (0.003)	0.925 (0.002)	1.067 (0.004)	0.857 (0.004)
	BPMF-SRIC	1.191 (0.004)	0.957 (0.004)	1.015 (0.004)	0.842 (0.004)
	HyPER	**1.173** (0.003)	**0.917** (0.002)	**1.001** (0.004)	**0.833** (0.004)

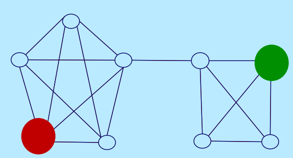




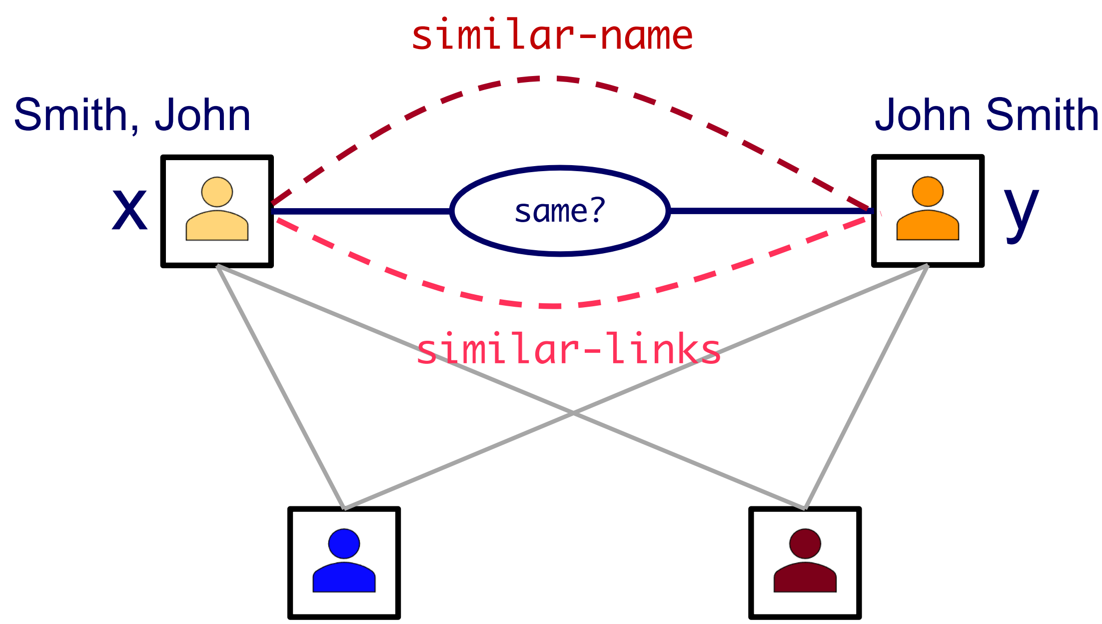
[RecSys] "HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems", Kouki, P., Fakhraei, S., Foulds, J., Eirinaki, M., & Getoor, L.

Bird's eye view

- Part 2: Complex and Heterogeneous Graphs
  - P 2.1: Factorization Methods
  - P 2.2: Heterogeneous Information Networks
  - P 3.3: Statistical Relational Learning
    - P3.3.1: Node Labeling / Collective Classification
    - P3.3.2: Link Prediction / Recommender Systems
    - P3.3.3: Entity Resolution / Knowledge Graph Identification



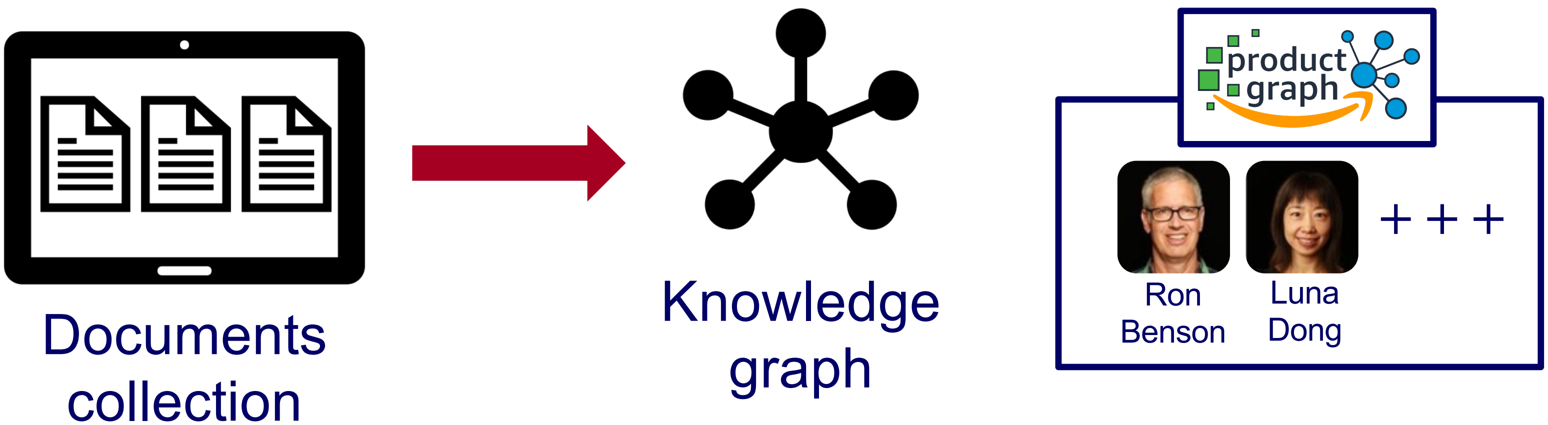
Entity Resolution




```
similar-name(x,y)  ->  same(x,y)
similar-links(x,y) ->  same(x,y)

same(x,y) & same(y,z) -> same(x,z)
```

Knowledge Graph Identification

How can we integrate noisy extracted facts into a knowledge graph?



We can:

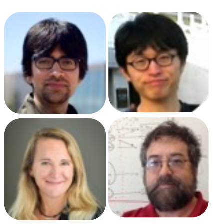
- Perform collective classification, entity resolution, link prediction
- Enforce ontological constraints
- Integrate different knowledge source information
- Combine them all!

Knowledge Graph Identification

```
// Ontological relations
             subsumes(l1,l2) &      label(e,l1) ->      label(e,l2)
            exclusive(l1,l2) &      label(e,l1) ->     ~label(e,l2)
              inverse(r1,r2) & relation(r1,e,o) -> relation(r2,o,e)
                 domain(r,l) &  relation(r,e,o) ->       label(e,l)
                  range(r,l) &  relation(r,e,o) ->       label(o,l)


// Entity resolution
         same-entity(e1,e2) &      label(e1,l) ->      label(e2,l)
         same-entity(e1,e2) & relation(r,e1,o) -> relation(r,e1,o)

// Integrating additional sources
                                label-nyt(e,l) ->       label(e,l)
                            label-youtube(e,l) ->       label(e,l)
                     relation-wikipedia(r,e,o) ->  relation(r,e,o)

// Sparsity
                                                 ~relation(r,e,o)
                                                      ~label(e,l)
```

Experimental Validation

	NELL		MusicBrainz		FreeBase	
	AUC	F1	AUC	F1	AUC	F1
MLN Ontology (Jiang, ICDM)	0.899	0.836				
Additional sources	0.888	0.843	0.672	0.788	0.416	0.734
Entity resolution	0.809	0.804	0.797	0.831		
Ontological relations	0.899	0.832	0.753	0.832	0.569	0.805
All of the above	**0.904**	**0.854**	**0.901**	**0.919**	**0.724**	**0.840**



[ISWC] "Knowledge graph identification", Pujara, J., Miao, H., Getoor, L., & Cohen, W.

[AI Magazine] "Using semantics and statistics to turn data into knowledge", Pujara, J., Miao, H., Getoor, L., & Cohen, W.

Software Tools

- PSL: Probabilistic soft logic
  https://psl.linqs.org/

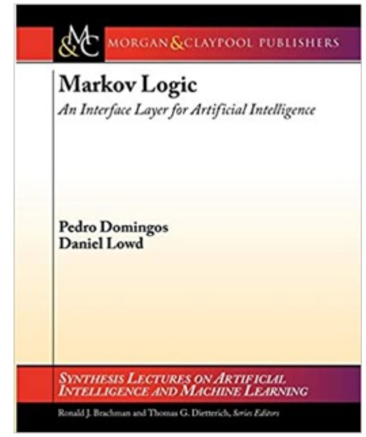
- Alchemy: Markov Logic Networks
  https://alchemy.cs.washington.edu/

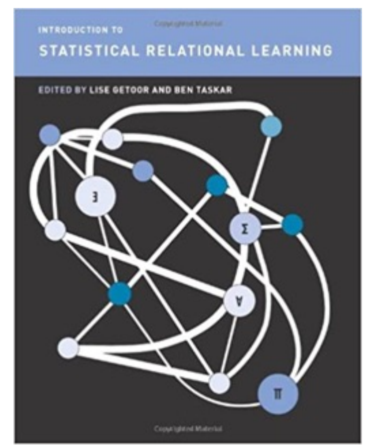
References

- Bach, Stephen H., Matthias Broecheler, Bert Huang, and Lise Getoor
  Hinge-loss markov random fields and probabilistic soft logic
  The Journal of Machine Learning Research, 2017

- Domingos, Pedro, and Daniel Lowd
  Markov logic: An interface layer for artificial intelligence
  Synthesis lectures on artificial intelligence and machine learning, 2009

- Lise Getoor, Ben Taskar (editors)
  Introduction to Statistical Relational Learning
  MIT Press, 2007





Bird's eye view

Task \ Tool	1.1 PR/HITS	1.1 PPR	1.2 METIS/ SVD	1.3 OddBall+	1.4 BP	2.1 FM	2.1 Tensor	2.2 HIN	2.3 SRL
1.1 Node Ranking	👍					(👍)		(👍)	👍
1.1' Link Prediction		👍				👍	(👍)	👍	👍
1.2 Comm. Detection			👍				👍	(👍)	(👍)
1.3 Anomaly Detection				👍			👍		
1.4 Propagation					👍			(👍)	👍

Part 1: Plain Graphs (columns 1.1 PR/HITS – 1.4 BP)

Part 2: Complex Graphs (columns 2.1 FM – 2.3 SRL)