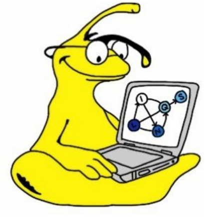
An Introduction to Probabilistic Soft Logic

Eriq Augustine and Golnoosh Farnadi
UC Santa Cruz

MLTrain 2018



psl.linqs.org
github.com/linqs/psl

Probabilistic Soft Logic (PSL) Overview

- Declarative probabilistic programming language for structured prediction
  - Scalable -- inference in PSL is highly efficient
  - Interpretable -- models are specified as weighted rules
  - Expressive -- can model complex dependencies, latent variables, handle missing data
- Open-source: psl.linqs.org

PSL Key Capabilities

- Rich representation language based on logic allows
    - Declarative representation of models
    - Well-suited to domains with structure (e.g., graphs and networks)
- Probabilistic Interpretation
    - Supports uncertainty and “soft” logic
    - Semantics defined via specific from of graphical model referred to as a *Hinge-loss Markov Random Field*

PSL Application Types

- Effective on wide range of problem types
    - data integration, information fusion, & entity resolution
    - recommender systems & user modeling
    - computational social science
    - knowledge graph construction

PSL Sample Application Domains

- Competitive Diffusion in Social Networks
  - Broecheler et al., SocialCom10
- Social Group Modeling
  - Huang et al., Social Networks and Social Media Analysis Workshop NIPS12
- Modeling Student Engagement in MOOCs
  - Ramesh et al., AAAI13; Ramesh et al., L@S14; Tomkins et al. EDM16
- Detecting Cyberbullying in Social Media
  - Tomkins et al., ASONAM
- Demographic Prediction & Knowledge Fusion for User Modeling
  - Farnadi et al., MLJ17
- Inferring Organization Attitudes in Social Media
  - Kumar et al., ASONAM16
- Personalization and Explanation in Hybrid Recommender Systems
  - Kouki et al., RecSys15; Kouki et al., RecSys17
- Drug-Drug Interaction
  - Sridhar et al, Bioinformatics 16

Outline

- Basic Introduction to PSL
- Getting Started with PSL
- PSL Examples
    - Collective Classification
    - Link Prediction
    - Entity Resolution
    - Knowledge Graph Construction
- Conclusion

Why Collective Classification?

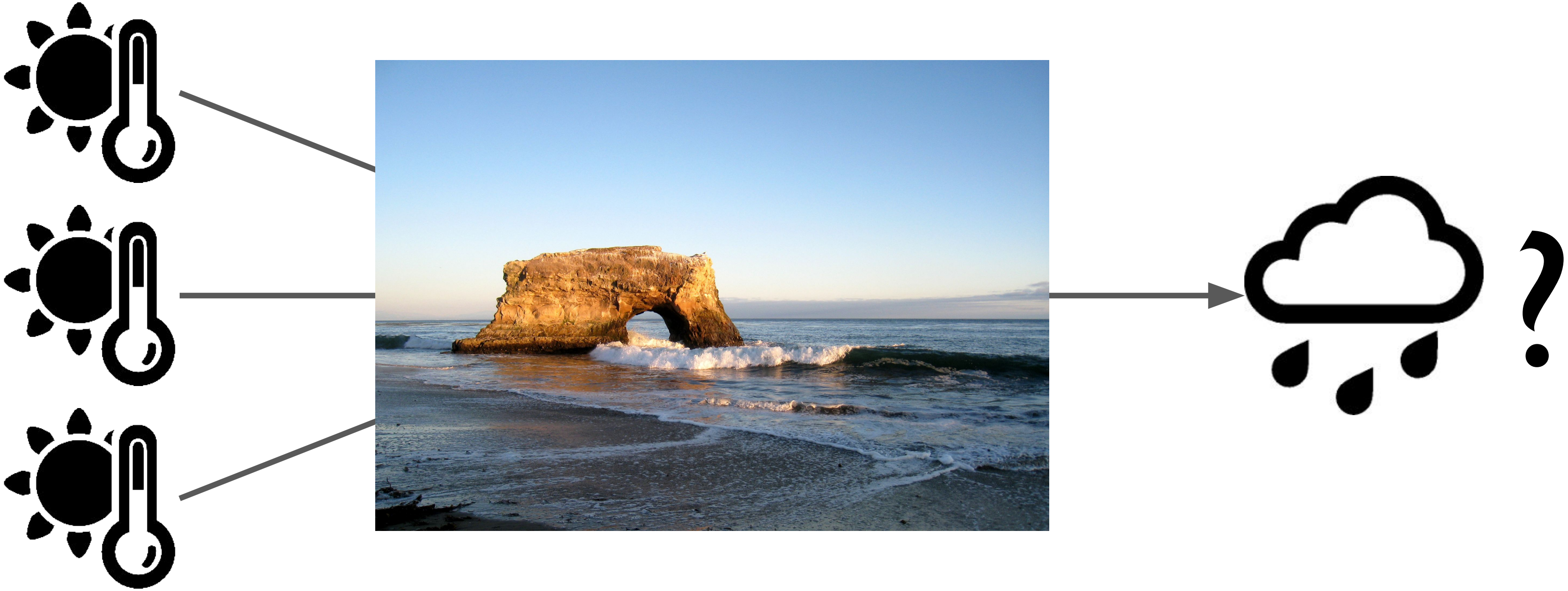
Weather Forecasting

Goal: Predict the probability of rain in Santa Cruz.

VS

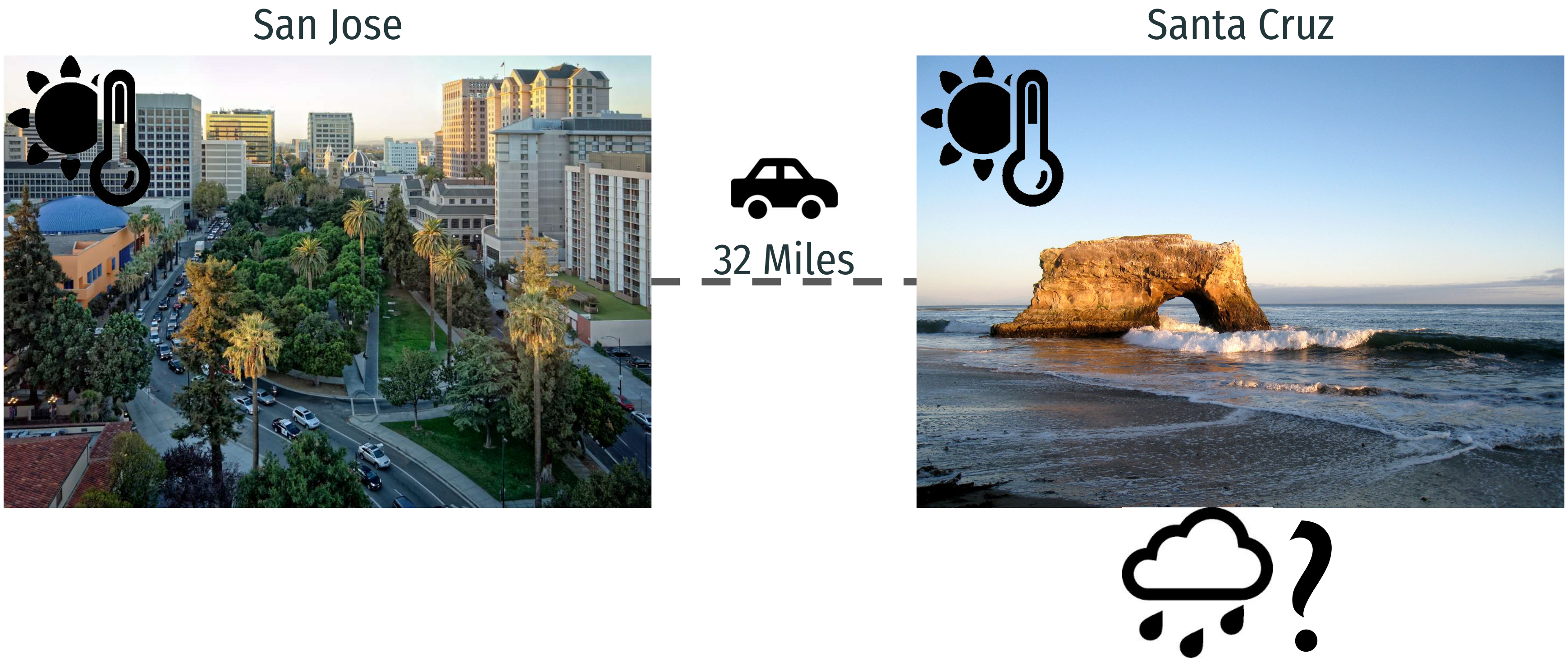
Local Signals for Prediction

Local sensors provide useful signals for prediction.



Relational Signals for Prediction

Sensors in nearby cities provide useful relational information.



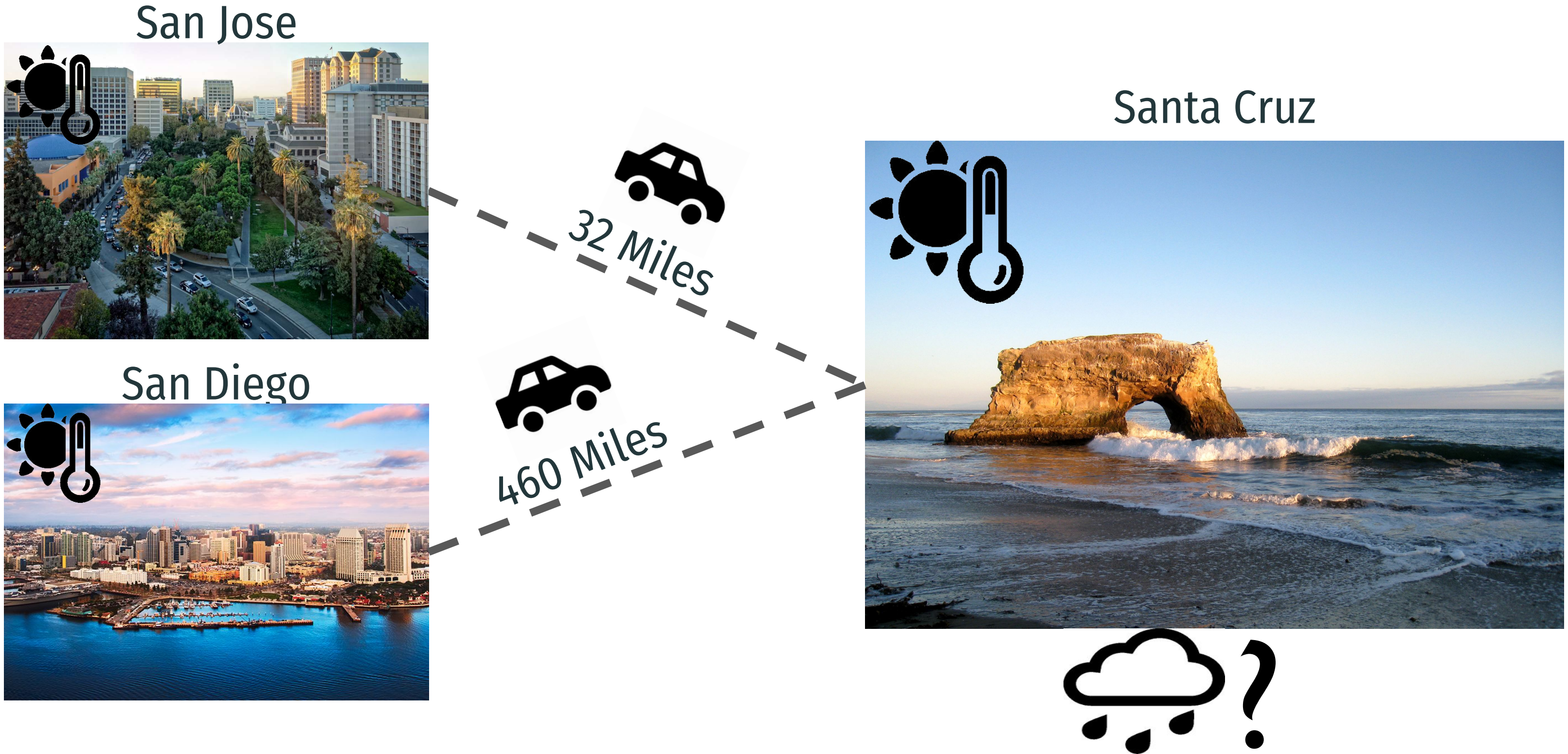
San Jose

32 Miles

Santa Cruz

Relational Signals for Prediction

Sensors in nearby cities provide useful relational information.



San Jose

San Diego

Santa Cruz

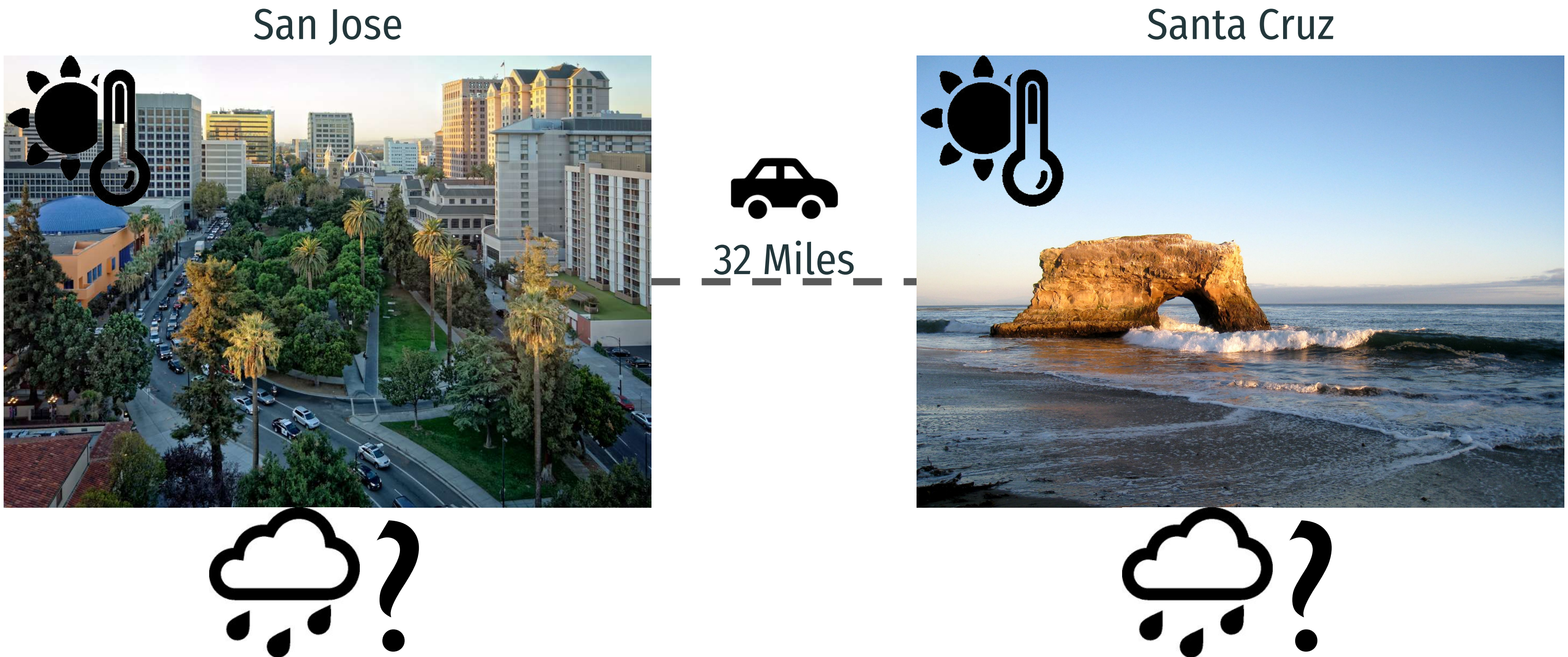
32 Miles

460 Miles

?

Weather Forecasting

What if we wanted to predict for multiple cities?



San Jose

32 Miles

Santa Cruz

Diagram for Weather Forecasting

sensor
Santa Cruz
San Jose
sensor

Diagram for Weather Forecasting

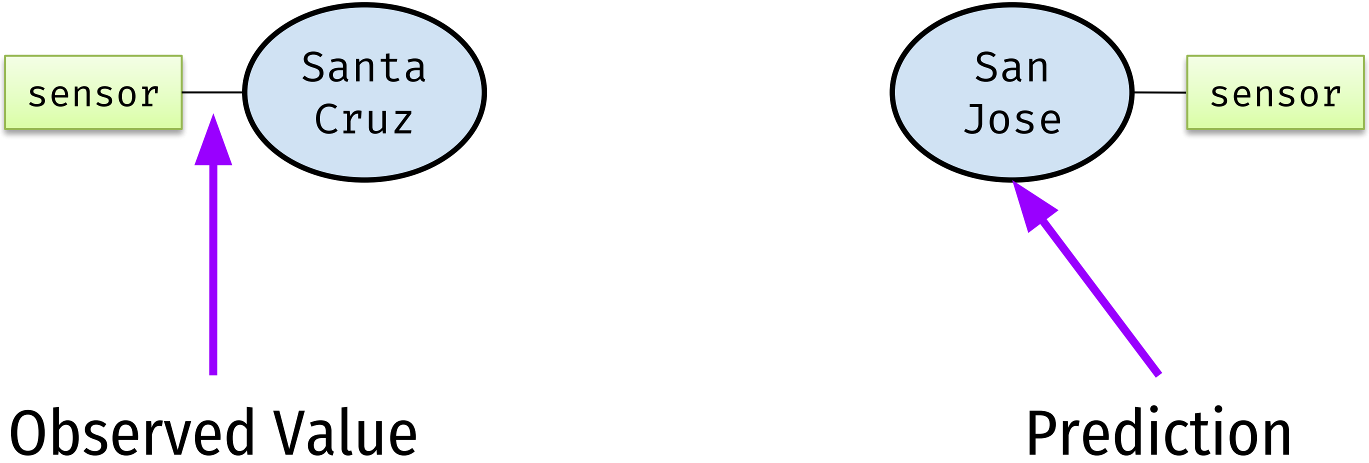
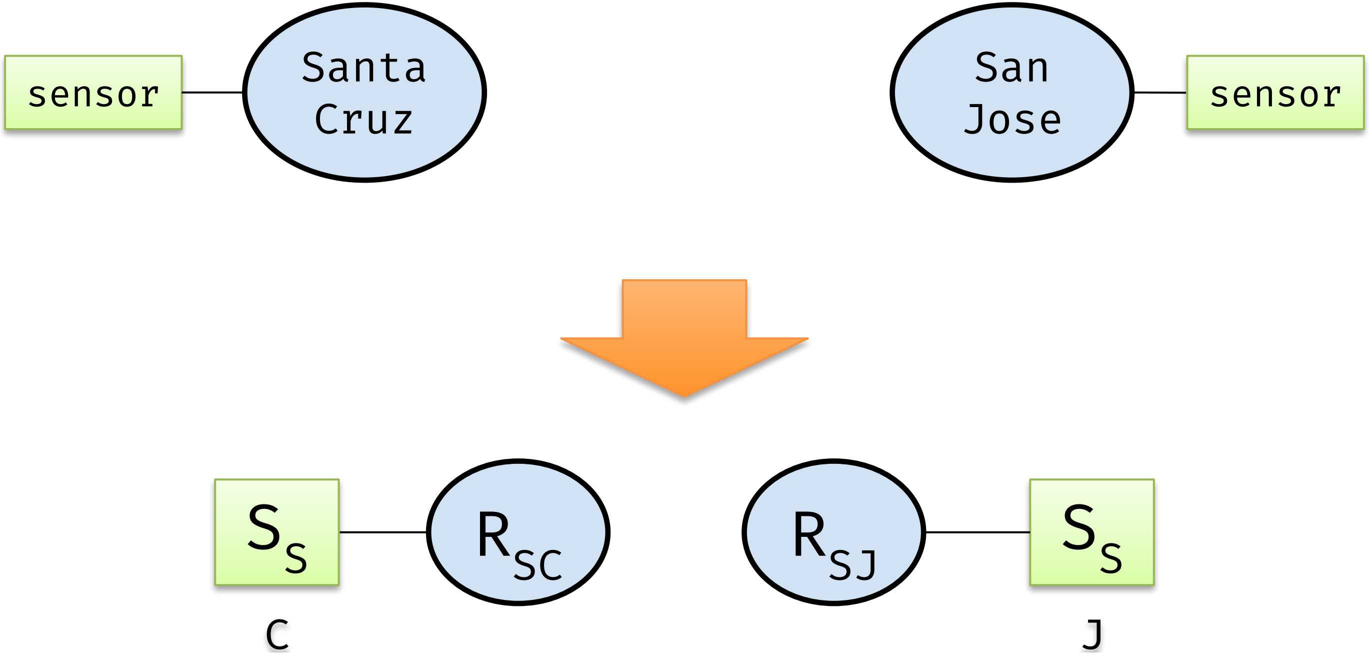
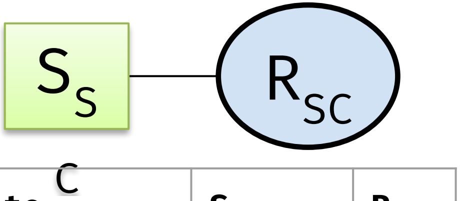


Diagram for Weather Forecasting



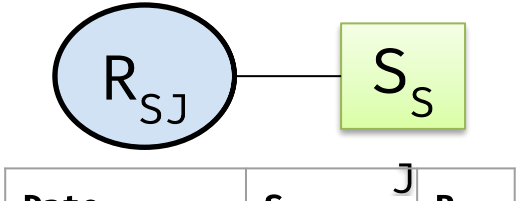
Local Predictive Model

Using historical data, we learn independent models for each city.



Date	**S_{SC}**	**R_{SC}**
1950-06-06	22.2°C	0
1951-06-06	17.1°C	1
...	...	...
2017-06-06	23.4°C	0

$$\Pr(R_{SC}|S_{SC})$$

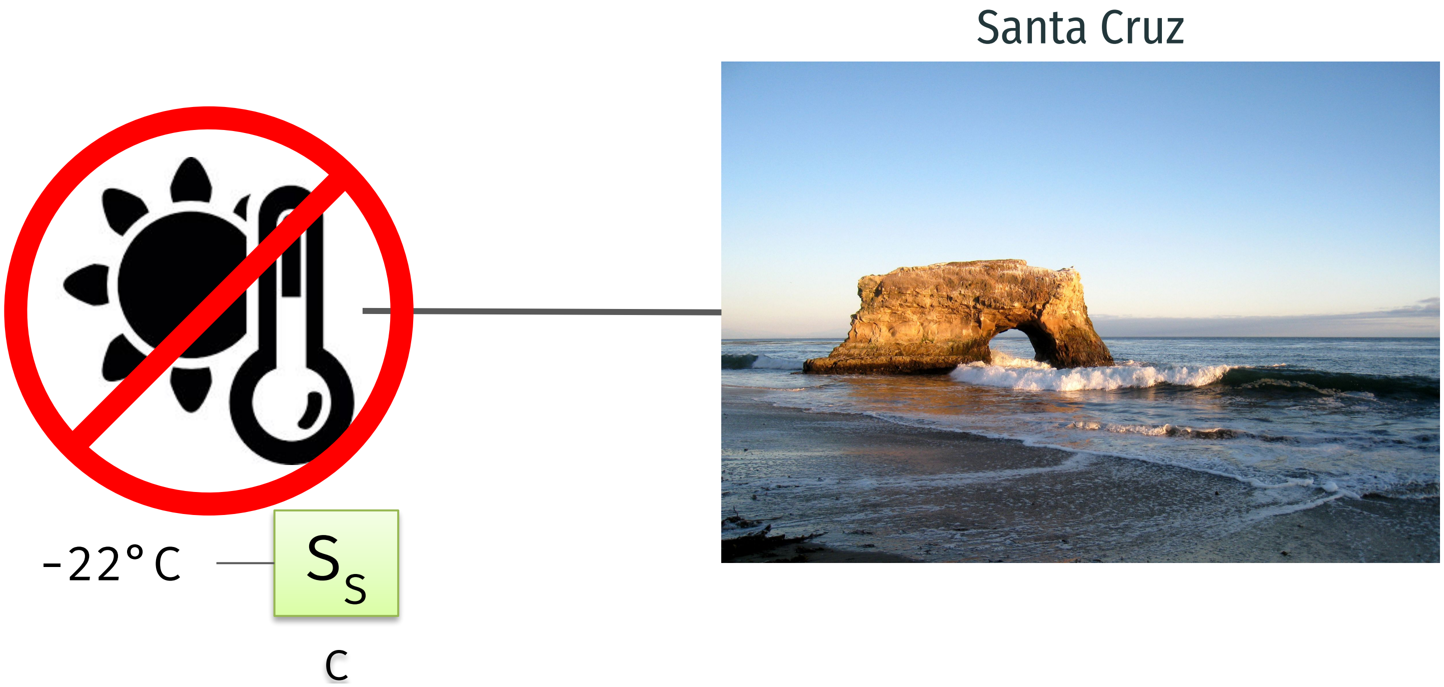


Date	**S_{SJ}**	**R_{SJ}**
1950-06-06	25.0°C	0
1951-06-06	20.1°C	1
...	...	...
2017-06-06	24.5°C	0

$$\Pr(R_{SJ}|S_{SJ})$$

Incorrect Sensor Reading

Common problem: we get a faulty sensor reading.



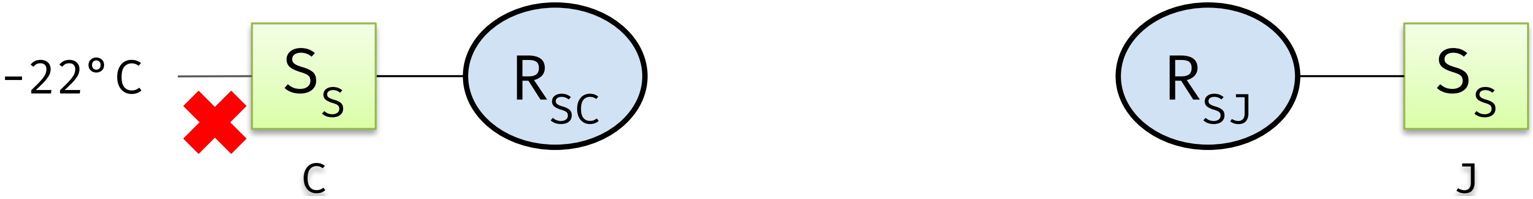
Santa Cruz

-22° C

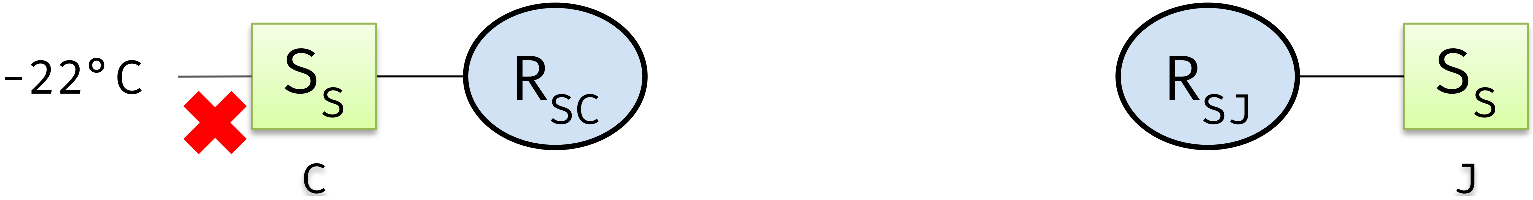
S_s

c

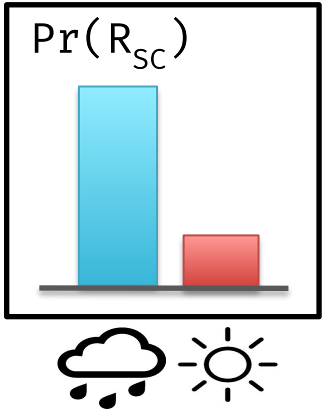
Incorrect Local Predictions



Incorrect Local Predictions

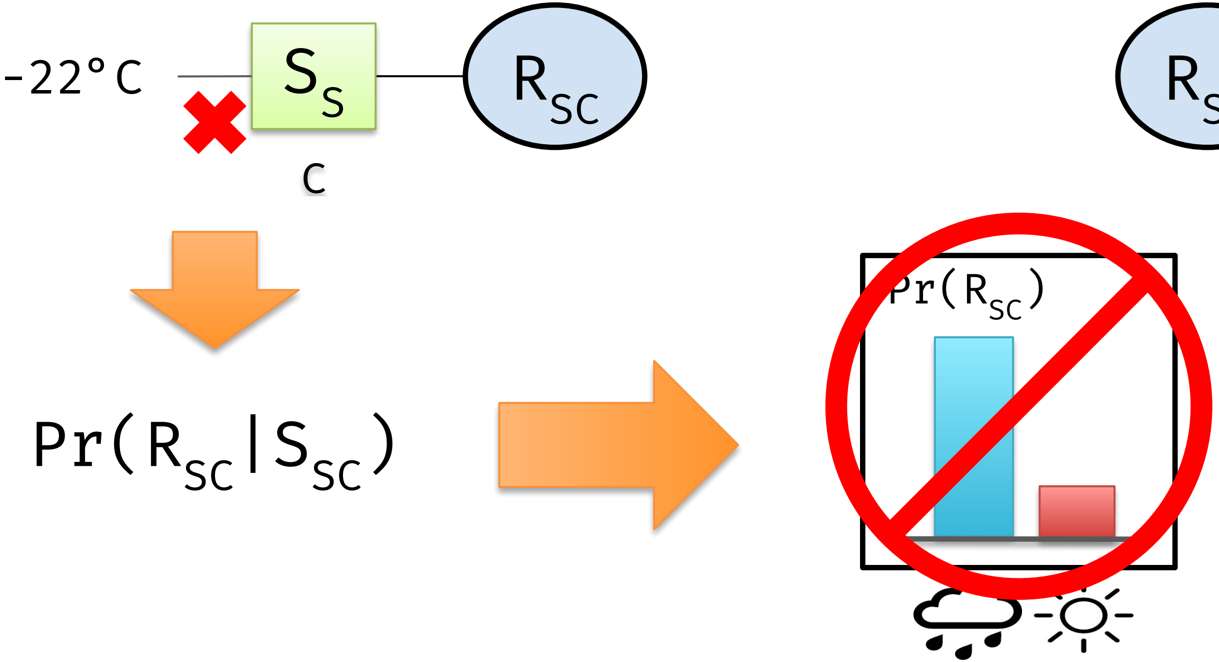


$Pr(R_{SC}|S_{SC})$



We use faulty reading to predict with our learned local model.

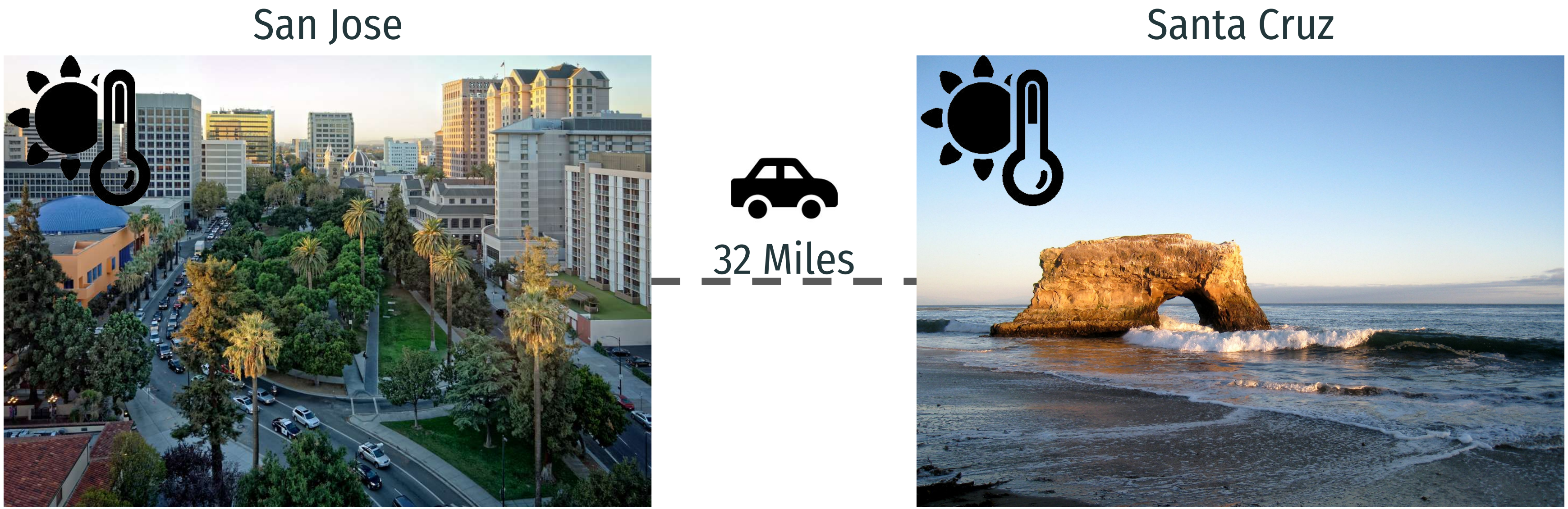
Incorrect Local Predictions



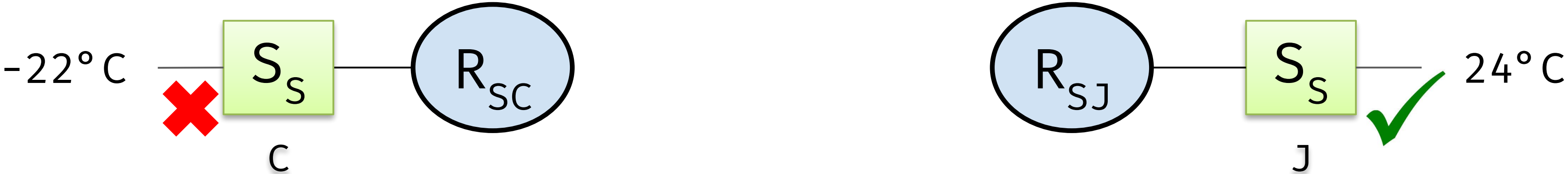
Common outcome: local model makes incorrect prediction.

Relational Signals for Prediction

Recall: sensors in nearby cities provide useful relational information!

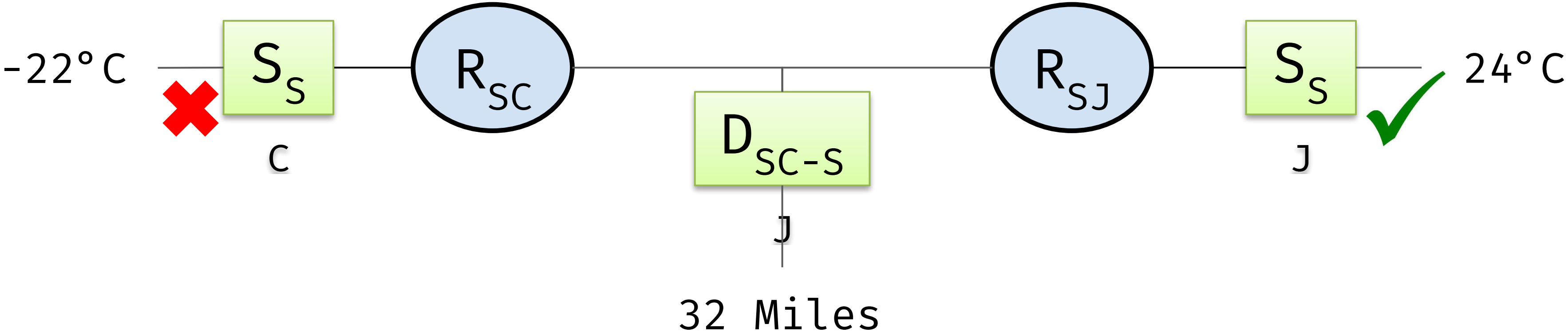


Leveraging Relational Signals



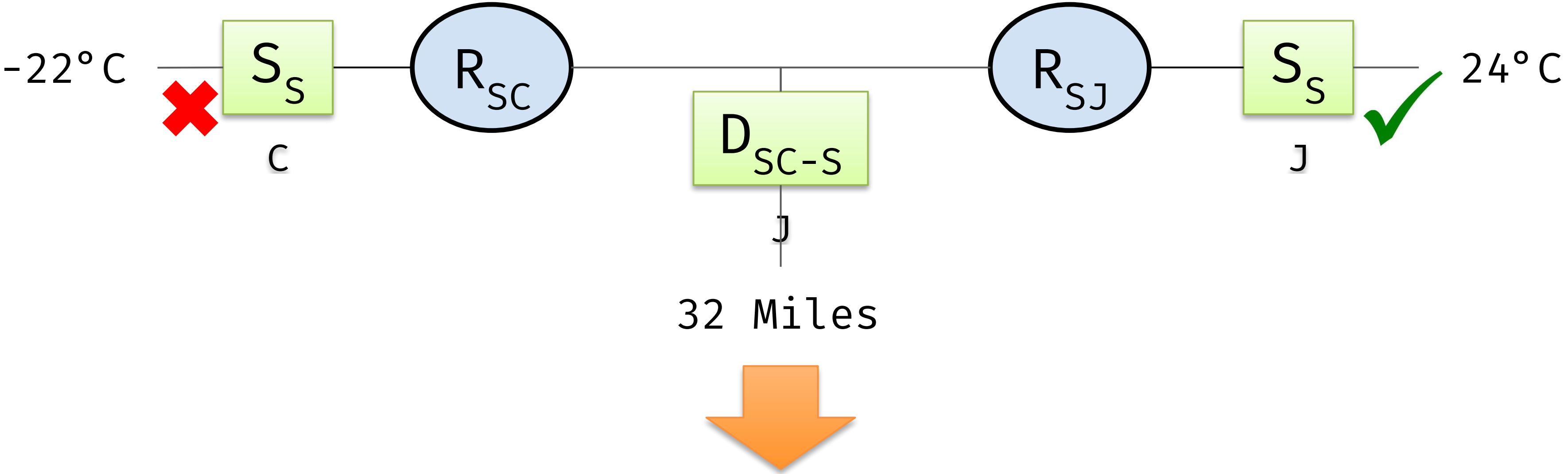
Leveraging Relational Signals

Distance variable captures closeness between cities.



Leveraging Relational Signals

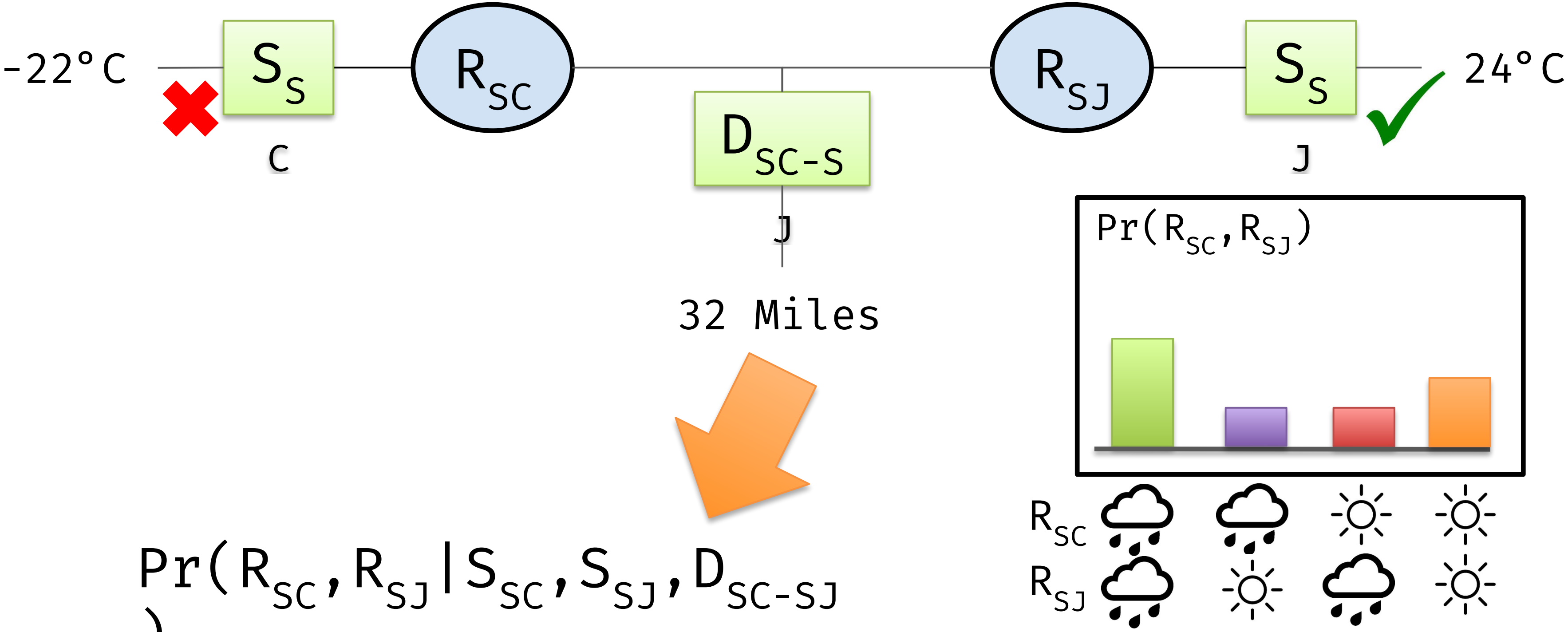
Distance variable captures closeness between cities.



$$Pr(R_{SC},R_{SJ}|S_{SC},S_{SJ},D_{SC-SJ})$$

Leveraging Relational Signals

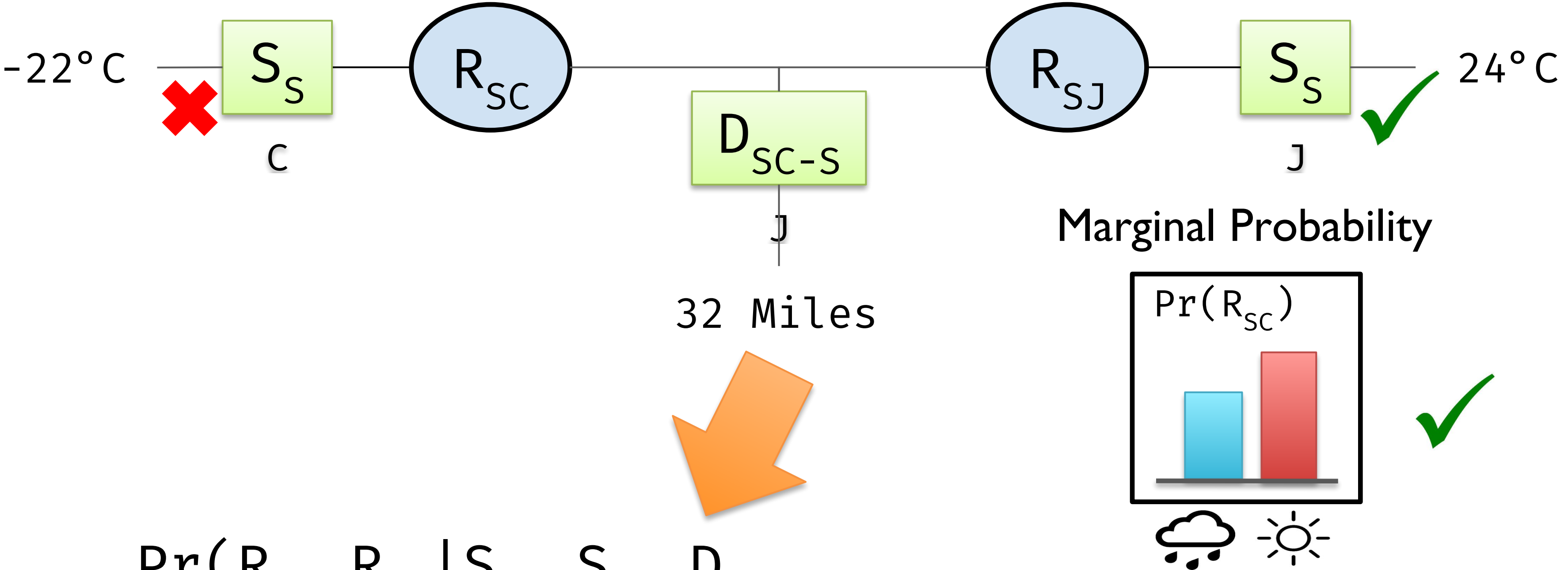
Joint modeling: forecasts in nearby cities should be similar.



$Pr(R_{SC}, R_{SJ} | S_{SC}, S_{SJ}, D_{SC-SJ})$

Leveraging Relational Signals

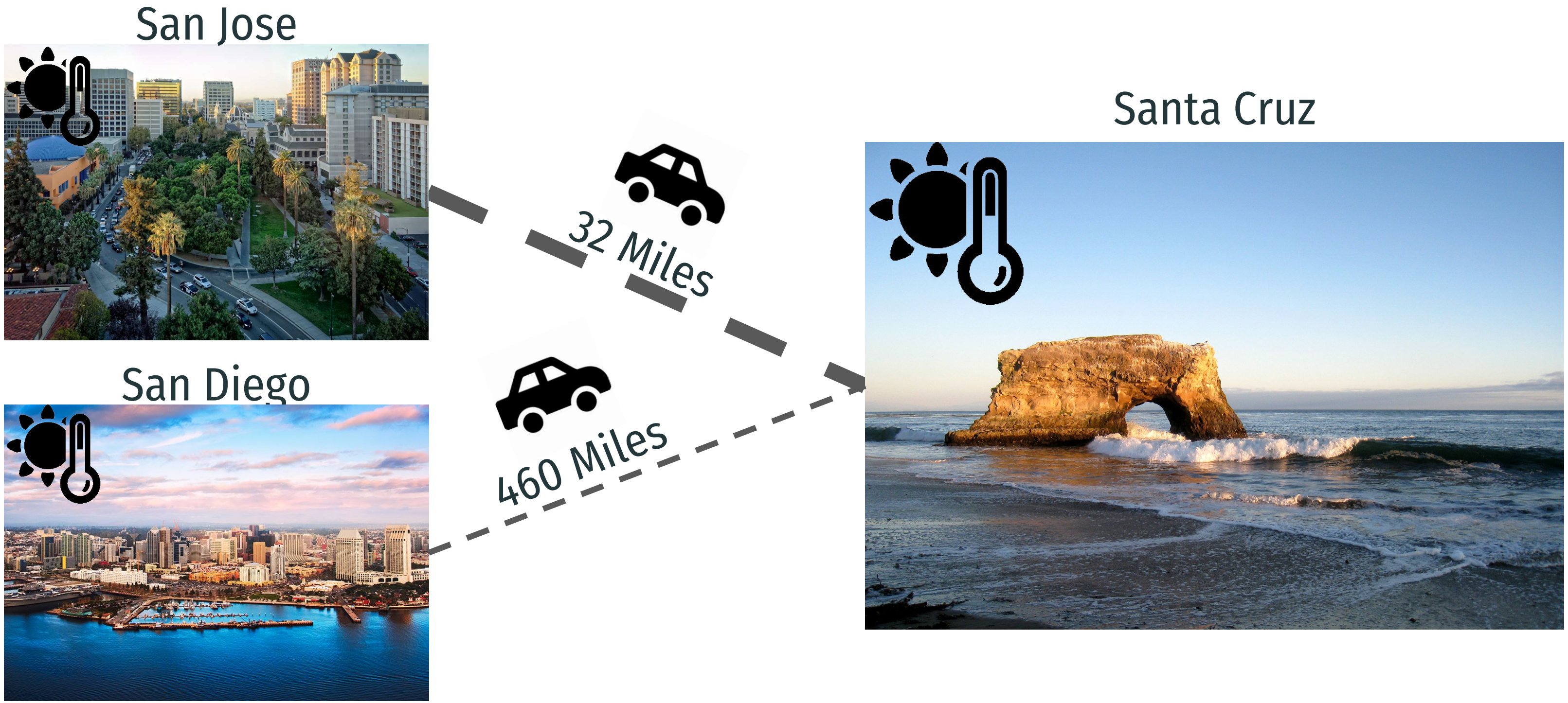
Joint modeling: forecasts in nearby cities should be similar.



$$Pr(R_{SC}, R_{SJ} | S_{SC}, S_{SJ}, D_{SC-SJ})$$

Combining Multiple Relational Signals

Nearby cities should have a greater relational influence than far away cities.



San Jose

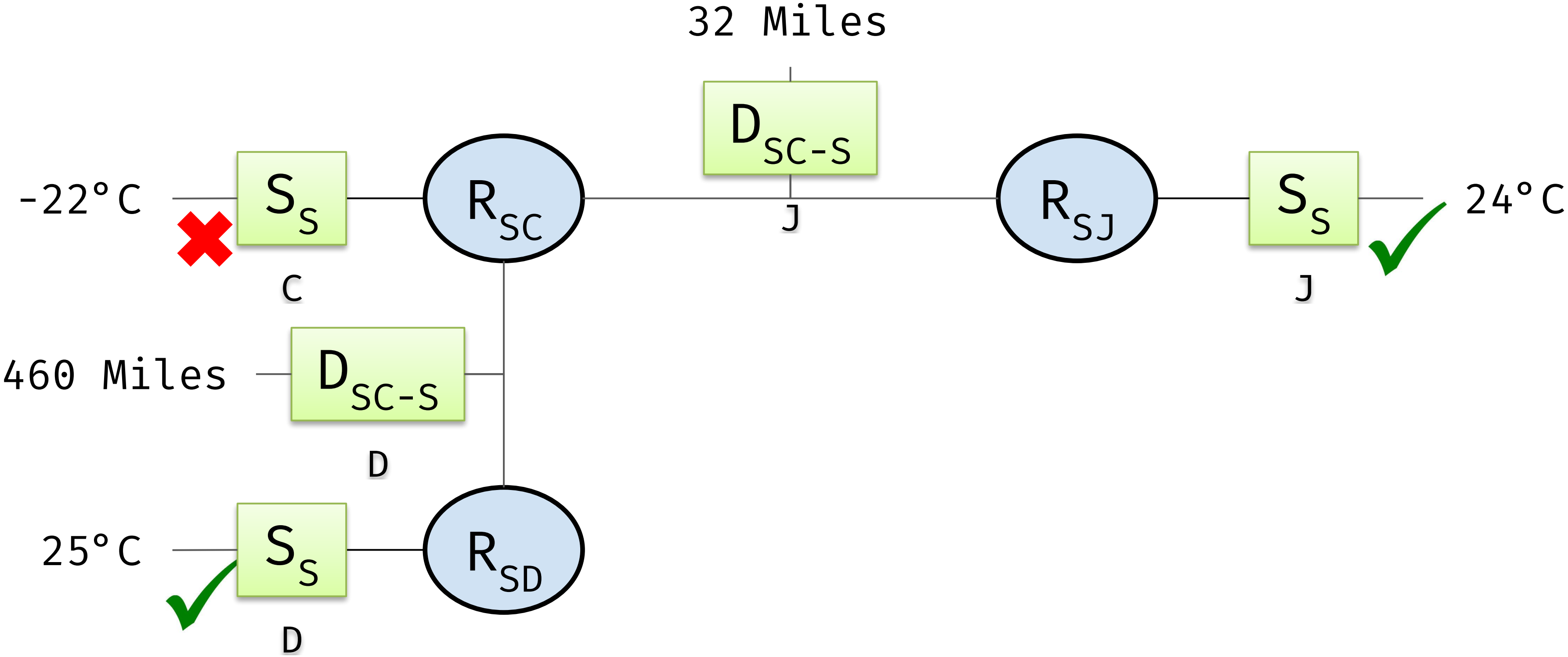
Santa Cruz

32 Miles

460 Miles

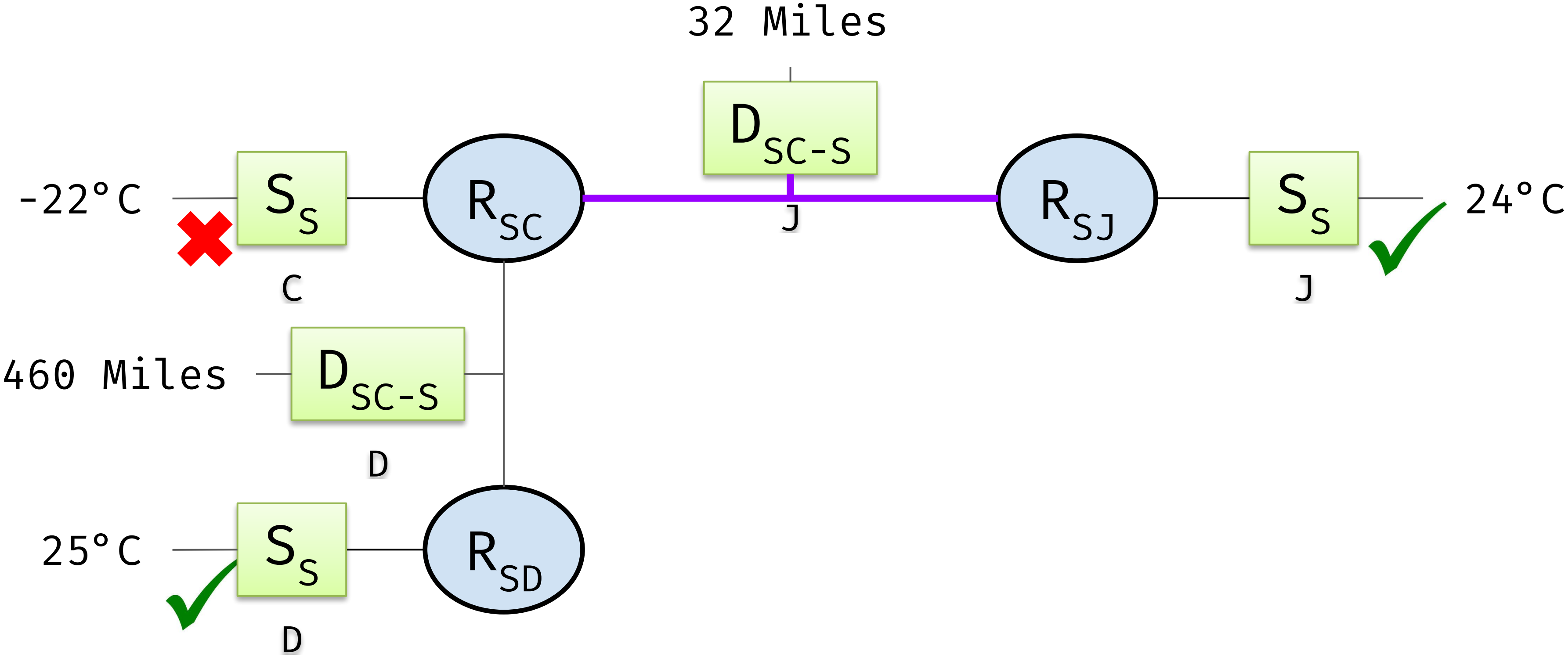
San Diego

Relative Influences of Neighbors



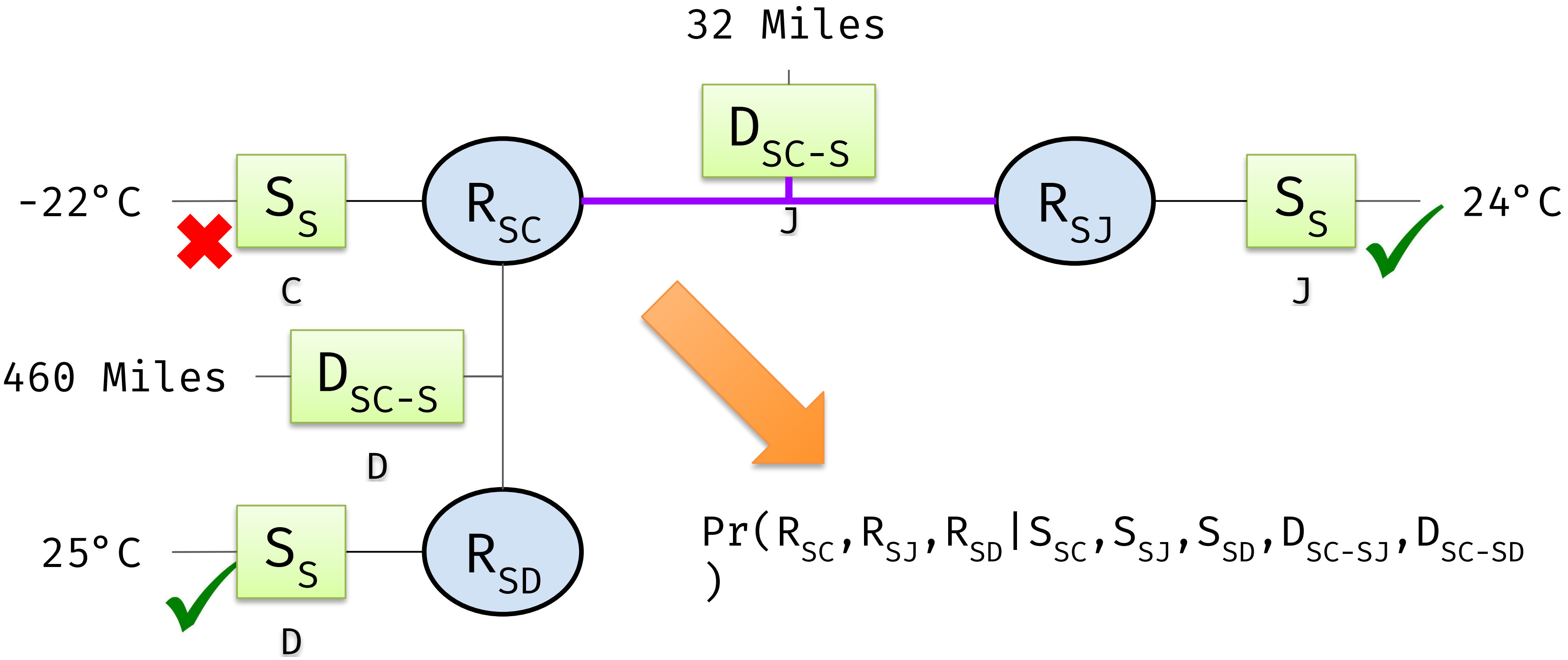
Relative Influences of Neighbors

Strength of collective influence depends on distance between cities.



Relative Influences of Neighbors

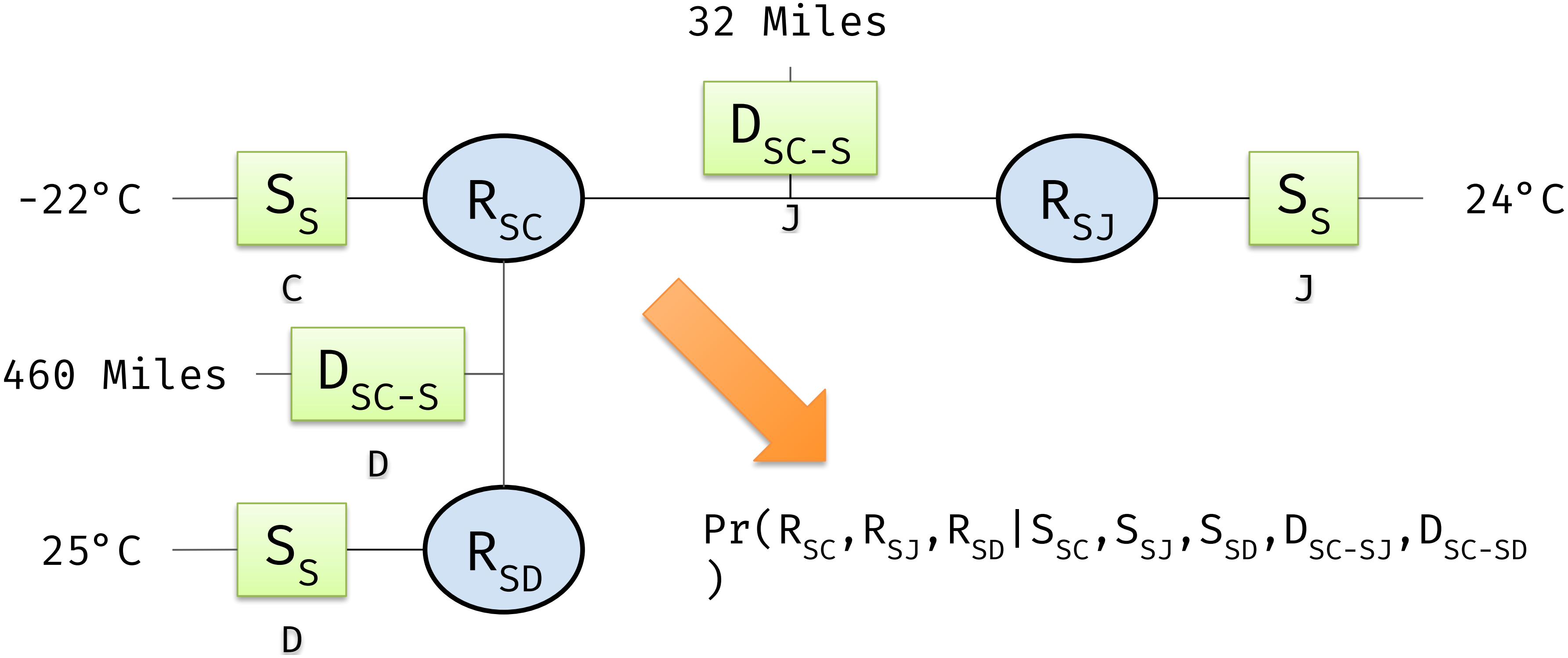
Distance variables $D_{SC\text{-}SJ}$ and $D_{SC\text{-}SD}$ mediate affinity of forecasts between cities.



$Pr(R_{SC},R_{SJ},R_{SD}|S_{SC},S_{SJ},S_{SD},D_{SC\text{-}SJ},D_{SC\text{-}SD}$
$)$

Markov Random Fields (MRFs)

This graphical model is a Markov Random Field (MRF).



$\mathrm{Pr}(R_{SC}, R_{SJ}, R_{SD} | S_{SC}, S_{SJ}, S_{SD}, D_{SC\text{-}SJ}, D_{SC\text{-}SD}$
$)$

PSL - Syntax and Semantics

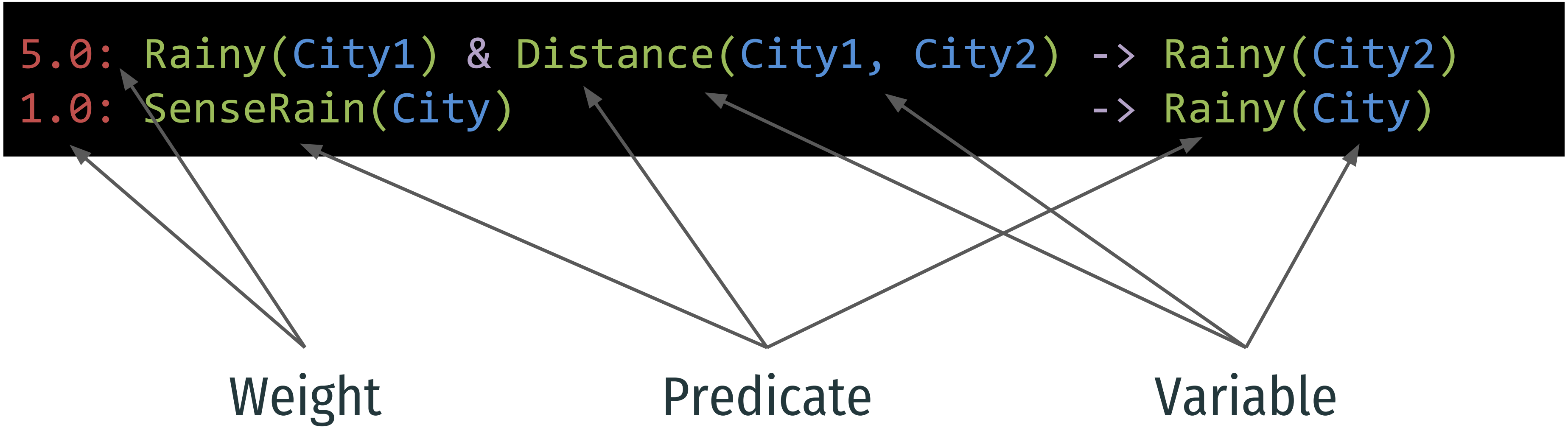
PSL

PSL uses first order logic-like rules.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
1.0: SenseRain(City)                       -> Rainy(City)
```

PSL

PSL uses first order logic-like rules.



```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
1.0: SenseRain(City)                       -> Rainy(City)
```

Weight

Predicate

Variable

PSL - Templating Language for MRFs

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

```
1.0: SenseRain(City) -> Rainy(City)
```

PSL - Templating Language for MRFs

Rule templates instantiated with data become "Ground Rules".

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

```
5.0: Rainy('Cruz') & Distance('Cruz', 'Jose') -> Rainy('Jose')
5.0: Rainy('Cruz') & Distance('Cruz', 'Diego') -> Rainy('Diego')
```

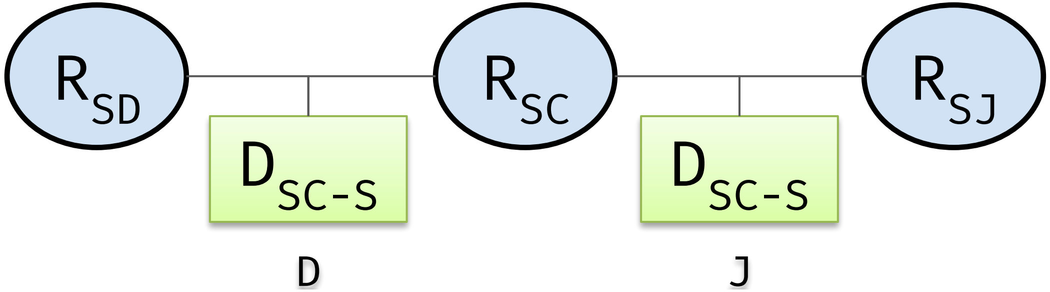
```
1.0: SenseRain(City) -> Rainy(City)
```

```
1.0: SenseRain('Cruz') -> Rainy('Cruz')
1.0: SenseRain('Jose') -> Rainy('Jose')
1.0: SenseRain('Diego') -> Rainy('Diego')
```

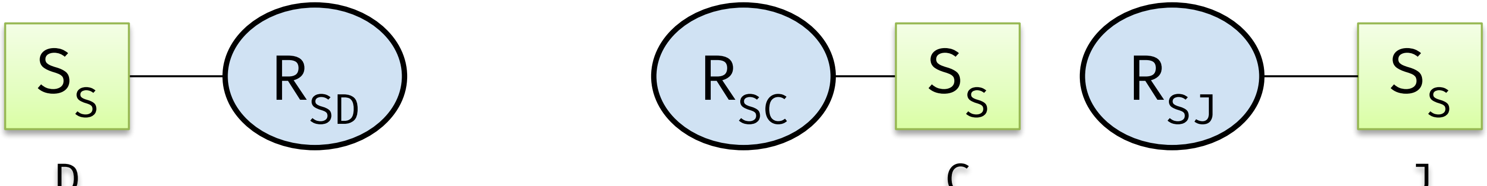
PSL - Templating Language for MRFs

Ground rules directly map to potential functions in the MRF.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

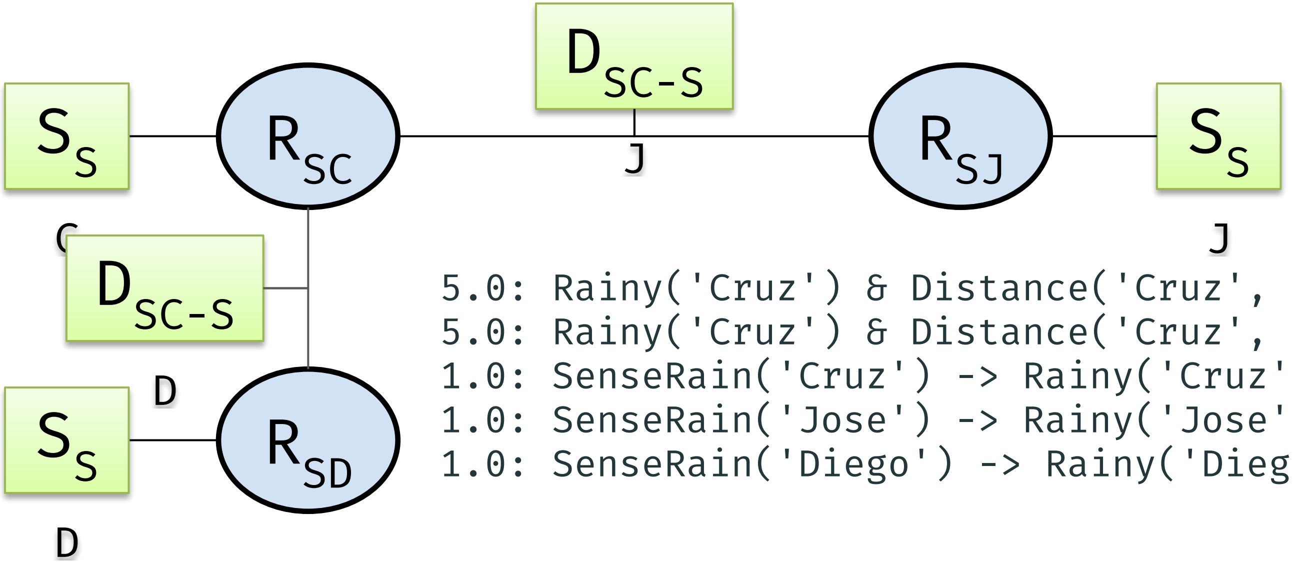


```
1.0: SenseRain(City)                        -> Rainy(City)
```



PSL - Templating Language for MRFs

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
1.0: SenseRain(City)                       -> Rainy(City)
```



```
5.0: Rainy('Cruz') & Distance('Cruz', 'Jose') -> Rainy('Jose')
5.0: Rainy('Cruz') & Distance('Cruz', 'Diego') -> Rainy('Diego')
1.0: SenseRain('Cruz') -> Rainy('Cruz')
1.0: SenseRain('Jose') -> Rainy('Jose')
1.0: SenseRain('Diego') -> Rainy('Diego')
```

PSL - MRF Inference

$$P(Y|X) \propto exp(\sum_{i}^{G} w_i\phi_i)$$

Sum over all ground rules.

The weight for a rule.

The "satisfaction" of a ground rule. 1/0 for discrete logic.

PSL - MRF Inference

$$P(Y|X) \propto exp(\sum_{i}^{G} w_i \phi_i)$$

$$\text{argmax}_X \sum_{i}^{G} w_i \phi_i$$

PSL - MRF Inference

Discrete MRF Inference == Weighted MAX-SAT == NP-Hard

$$\mathrm{argmax}_X \sum_{i}^{G} w_i \phi_i$$

PSL - Continuous Relaxation

Relax "hard" satisfiability of each rule.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

PSL - Continuous Relaxation

First convert the rule to Disjunctive Normal Form.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

Rainy(City1) ^ Distance(City1, City2) -> Rainy(City2)

¬(Rainy(City1) ^ Distance(City1, City2)) v Rainy(City2)

¬Rainy(City1) v ¬Distance(City1, City2) v Rainy(City2)

PSL - Continuous Relaxation

Use Łukasiewicz logic to relax hard logical operators.

- `P ^ Q = max(0.0, P + Q - 1.0)`
- `P v Q = min(1.0, P + Q)`
- `¬Q = 1.0 - Q`

PSL - Continuous Relaxation

Apply Łukasiewicz logic.

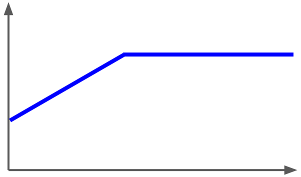
```
¬Rainy(City1) v ¬Distance(City1, City2) v Rainy(City2)

min(1.0, ¬Rainy(City1) + ¬Distance(City1, City2)) v Rainy(City2)

min(1.0, ¬Rainy(City1) + ¬Distance(City1, City2) + Rainy(City2)

min(1.0, (1.0 - Rainy(City1)) + (1.0 - Distance(City1, City2))
    + Rainy(City2))

min(1.0, 2.0 - (Rainy(City1) + Distance(City1, City2))
    + Rainy(City2))
```

PSL - Continuous Relaxation

Apply Łukasiewicz logic to form a Hinge-Loss MRF.

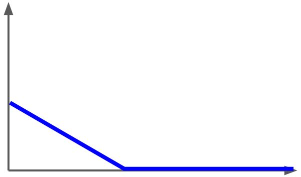
Satisfaction:

```
min(1.0, 2.0 - (Rainy(City1) + Distance(City1, City2)) + Rainy(City2))
```



Distance to satisfaction:

```
1.0 - min(1.0, 2.0 - (Rainy(City1) + Distance(City1, City2)) + Rainy(City2))
```



PSL - HL-MRF Inference

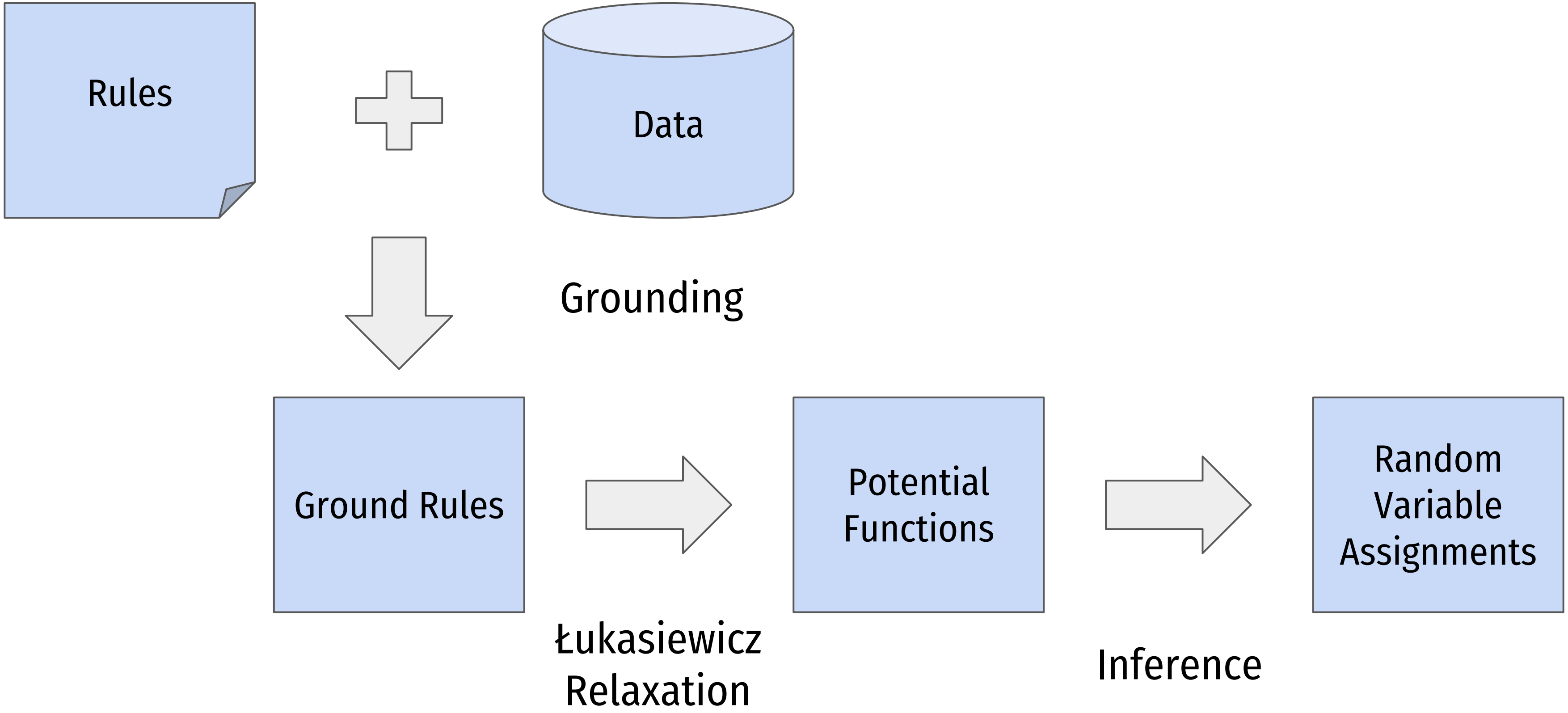
HL-MRF Inference == Sum of Convex Function == Convex!

Solve with Alternating Direction Method of Multipliers (ADMM)

$$\text{argmax}_X \sum_{i}^{G} w_i \phi_i$$

https://web.stanford.edu/~boyd/admm.html

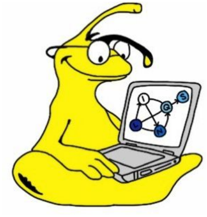
PSL - Rules to Assignments



Getting Started with PSL

Eriq Augustine and Golnoosh Farnadi
UC Santa Cruz

MLTrain 2018



psl.linqs.org

github.com/linqs/psl

Getting the Code

```
git clone https://github.com/linqs/psl-examples.git

cd psl-examples/simple-acquaintances/cli

git checkout uai18
```

All examples available at: https://github.com/linqs/psl-examples

Requirements

CLI:

- Java 7/8

Java/Groovy:

- Java 7/8
- Maven

Helper Scripts:

- Linux / Mac / Windows Subsystem for Linux
- wget / curl

Toy Problem

- Predict who knows who.
- Given information:
  - Where people have lived.
  - What people like.
  - Who some people already know.

What a PSL Example Looks Like

```
simple-acquaintances
├── README.md
├── cli
│   ├── run.sh
│   ├── simple-acquaintances.data
│   └── simple-acquaintances.psl
├── data
└── groovy
    ├── pom.xml
    ├── run.sh
    └── src
```

Examining the Model

- CLI PSL requires two files:
  - Model/Rules File
    - Defines Rules
  - Data File
    - Defines Predicates
    - Defines Partitions
    - Points to Actual Data

Running a PSL Example

```
./run.sh
```

Performed by the run script:

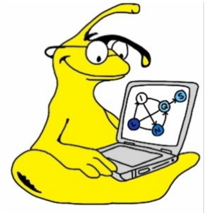
- Fetch Data
- Fetch PSL Dependencies
- Build
- Run Weight Learning
- Run Inference
- Evaluate Results
- Output Predictions

Configuring PSL

- CLI Usage
- Modifying Run Script
- Configuration Options
  - Logging
  - Postgres
  - Inference Hyperparms
  - Lazy Inference
- Weight Learning
  - Different Methods

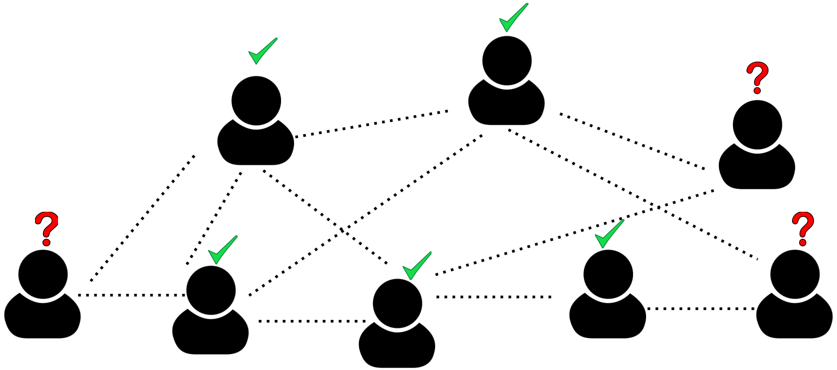
Collective Classification

Eriq Augustine and Golnoosh Farnadi
UC Santa Cruz
MLTrain 2018

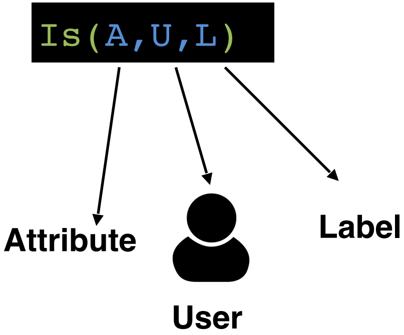


psl.linqs.org
github.com/linqs/psl

What is Collective Classification?

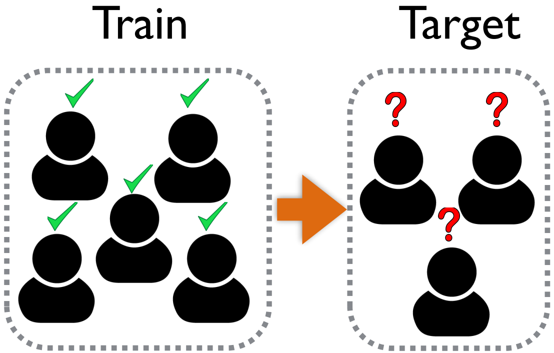


Attribute "**A**" of User "**U**" is Labeled "**L**"



```
Is(A,U,L)
```

What is Collective Classification?



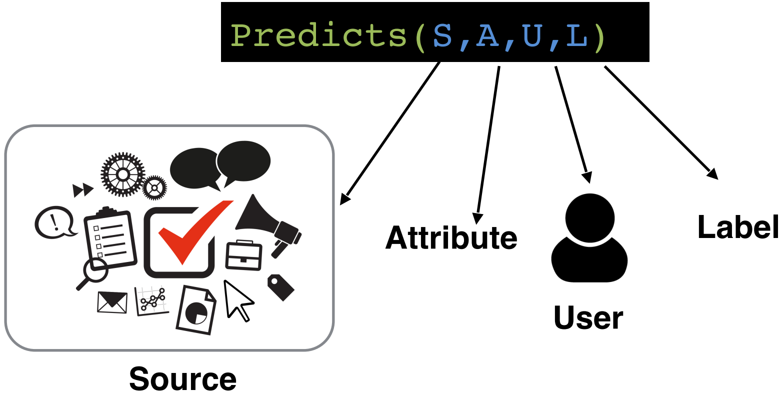
example:

```
Is(Gender,Alice,Female)
```

```
Is(Age,Bob,Young)
```

```
Is(Personality,Carol,Introvert)
```

Local Predictor Rule



```
Predicts(S,A,U,L)
```

Attribute

User

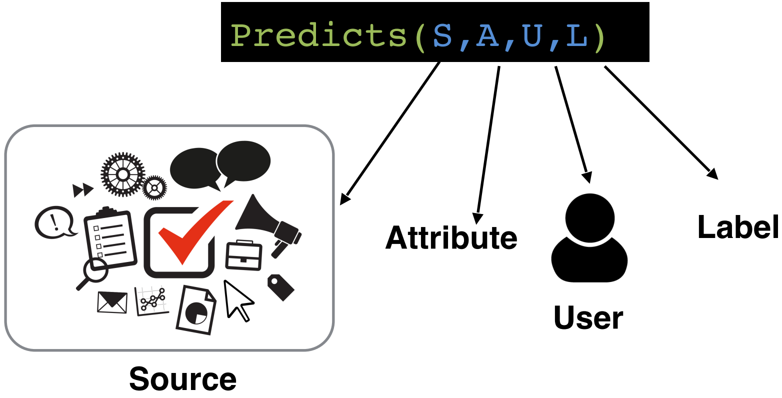
Label

Source

Source “**S**” **Predicts** Attribute “**A**” of User “**U**” **is** Labeled “**L**”

```
Predicts(S,A,U,L) -> Is(A,U,L)
```

Local Predictor Rule



`Predicts(S,A,U,L)`

Source

Attribute

User

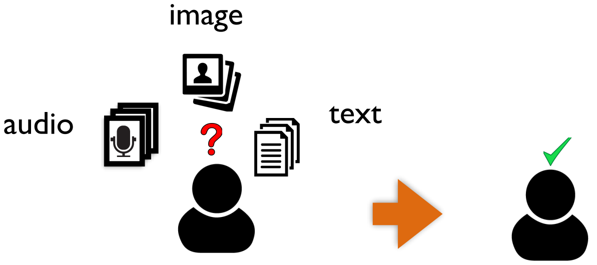
Label

We collect training data to learn a predictive model, e.g. logistic regression

T_U	L_U
🗐	0
⋮	⋮
🗐	1

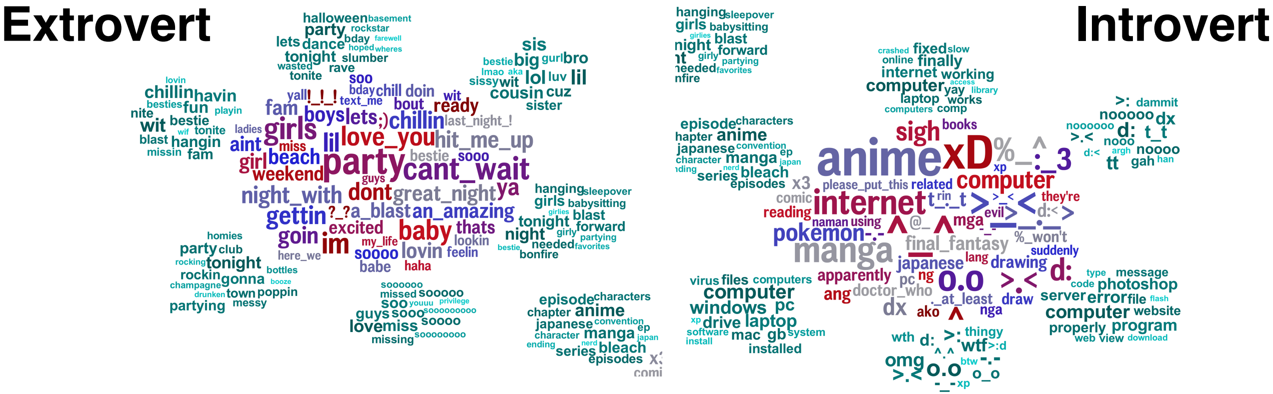
$$P(L|T)$$

Local Predictor Rule



example:

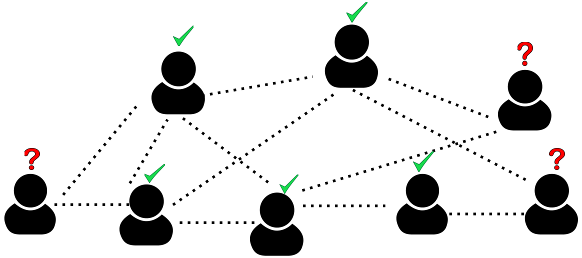
```
Predicts(Txt,Personality,Alice,Int)
                         -> Is(Personality,Alice,Int)
```



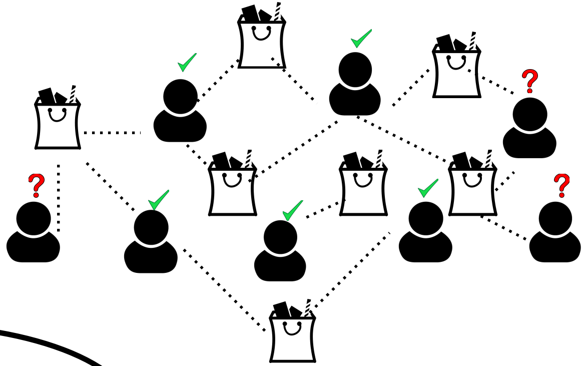
Park G, Schwartz HA, Eichstaedt JC, Kern ML, Kosinski M, Stillwell DJ, Ungar LH, & Seligman ME (2014). Automatic Personality Assessment Through Social Media Language. Journal of Personality and Social Psychology

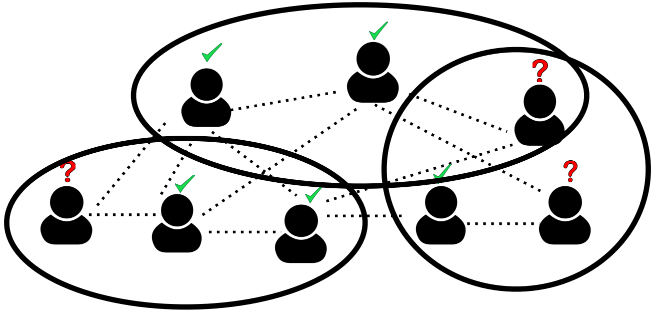
Collective Rule

(user-user relations)



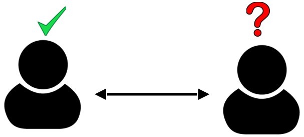
(user-item relations)





(user-group relations)

Collective Rule (1/3) (user-user relations)

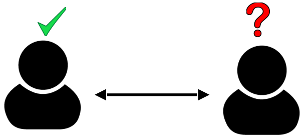


- Friend
- Follower
- Neighbour
- Spouse
- Idol
- Coauthor
- Colleague

u1	u2
u2	u3
…	…
un	u22

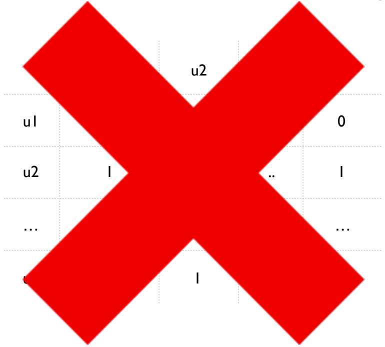
	u1	u2	…	un
u1	1	1		0
u2	1	1	..	1
…	…	…		…
un	0	1	…	1

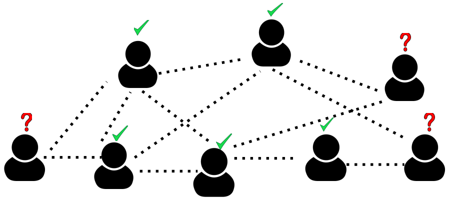
Collective Rule (1/3) (user-user relations)



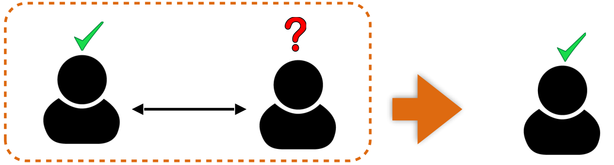
- Friend
- Follower
- Neighbour
- Spouse
- Idol
- Coauthor
- Colleague

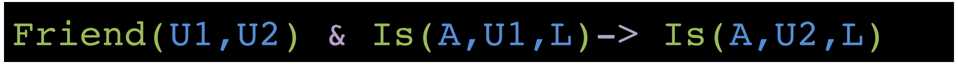
u1	u2
u2	u3
…	…
un	u22


u2
u1
0
u2
1
..
1
…
…
1



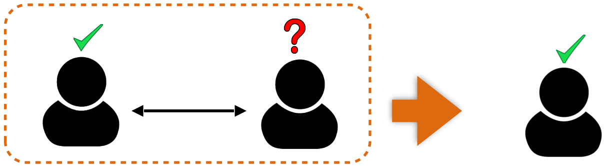
Collective Rule (1/3) (user-user relations)





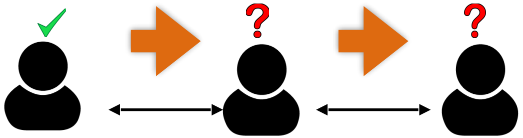
```
Friend(U1,U2) & Is(A,U1,L)-> Is(A,U2,L)
```

Collective Rule (1/3) (user-user relations)

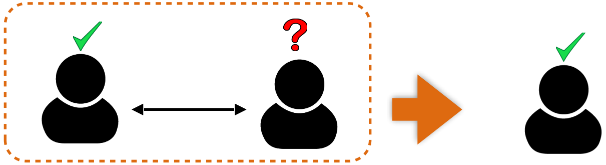


```
Friend(U1,U2) & Is(A,U1,L) -> Is(A,U2,L)
```

Open predicate



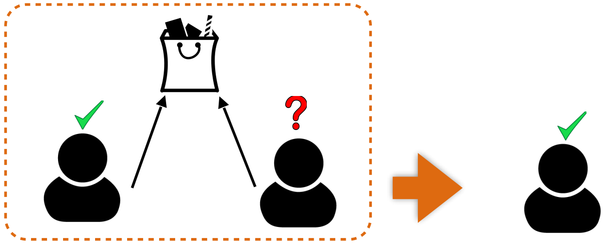
Collective Rule (1/3) (user-user relations)



example:

```
Friend(Alice,Carol) & Is(Personality,Carol,Ext)
                          -> Is(Personality,Alice,Ext)
```

Collective Rule (2/3) (user-item relations)



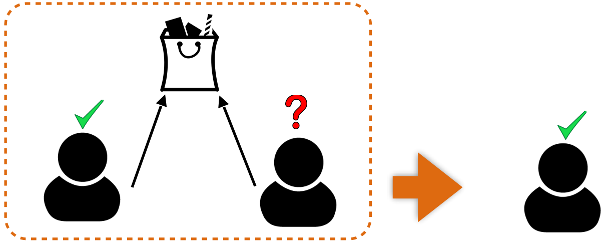
- Page likes
- Item rating
- Movie ratings

u1	I1
u2	I7
…	…
un	I8

	I1	I2	…	In
u1	1	1		0
u2	1	1	..	1
…	…	…		…
un	0	1	…	1

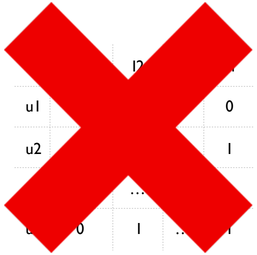
matrix-factorisation

Collective Rule (2/3) (user-item relations)

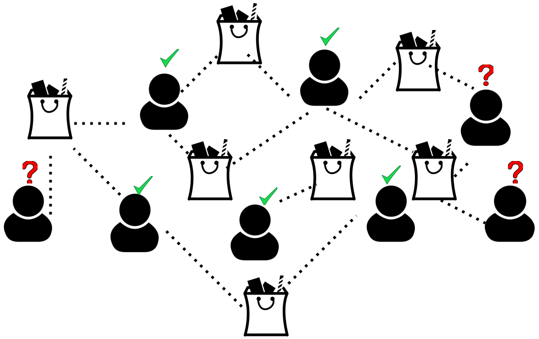


- Page likes
- Item rating
- Movie ratings

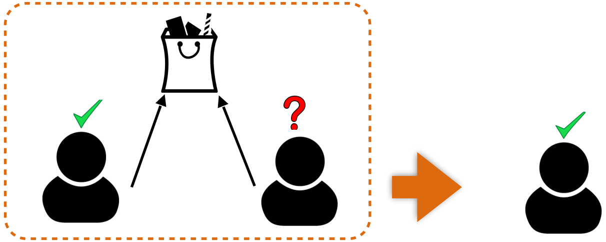
u1	I1
u2	I7
…	…
un	I8


I2
u1
0
u2
1
…
0
1
…

matrix-factorisation

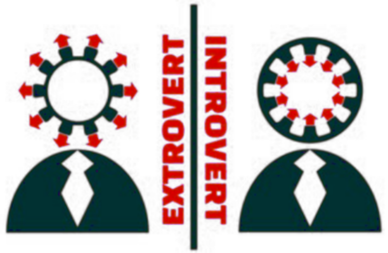


Collective Rule (2/3) (user-item relations)



```
Likes(U1,I) & Likes(U2,I) & Is(A,U1,L)-> Is(A,U2,L)
```

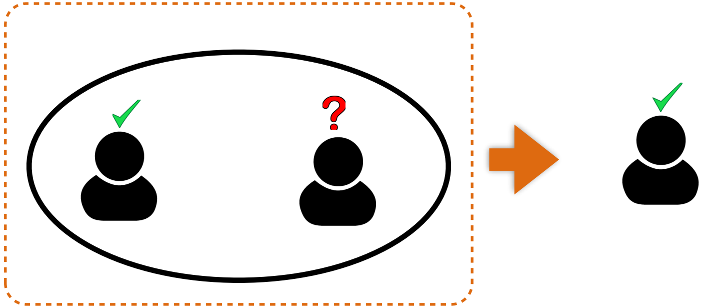
Collective Rule (2/3) (user-item relations)



example:

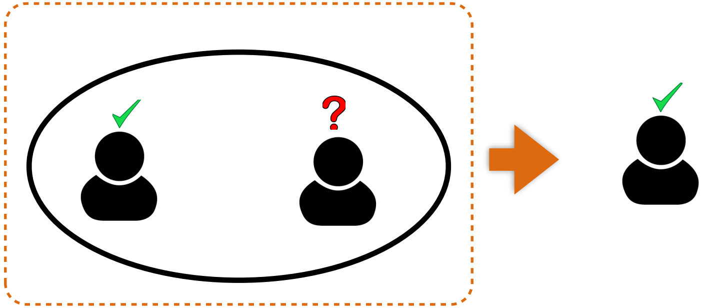
```
Likes(Alice,Partying) & Likes(Carol,Partying) &
     Is(Personality,Carol,Ext)-> Is(Personality,Alice,Ext)
```

Collective Rule (3/3) (user-group relations)



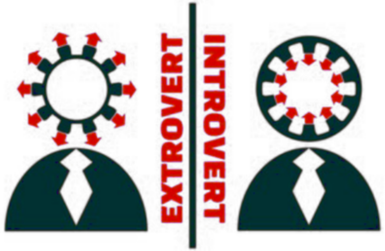
- groups
- clusters

Collective Rule (3/3) (user-group relations)



```
Joins(U1,G) & Joins(U2,G) & Is(A,U1,L)-> Is(A,U2,L)
```

Collective Rule (3/3) (user-group relations)



example:

```
Joins(Carol,Action-Movies) & Joins(Alice,Action-Movie) &
     Is(Personality,Carol,Ext)-> Is(Personality,Alice,Ext)
```

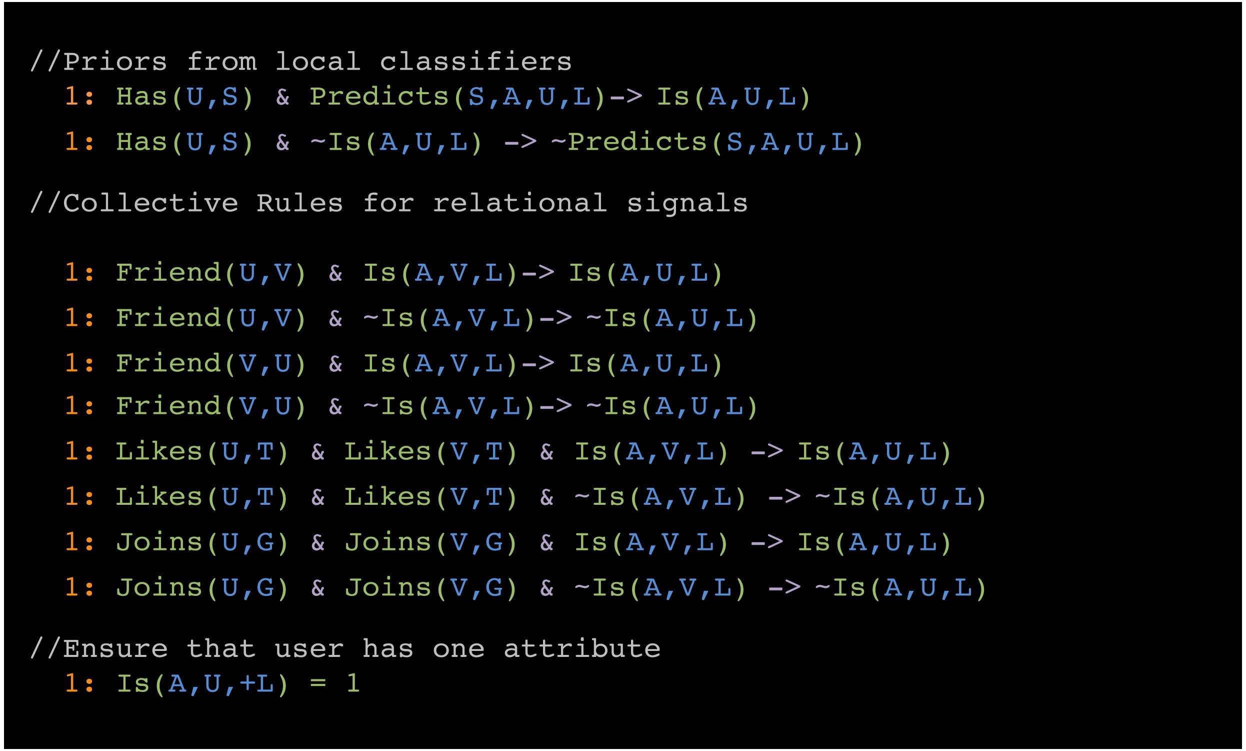
Hands on

- **Data**: Synthetic data, friendship links is a network whose degree distribution follows a power law, with 100 users, two local predictors, one set of joins relations and one set of likes relations.

- git clone https://github.com/linqs/psl-examples.git
- cd psl-examples/user-modeling/cli
- git checkout uai18

- Models
  - Local predictor (Text and Image)
  - Friendship
  - Likes
  - Joins
  - all

PSL Model for User Modeling

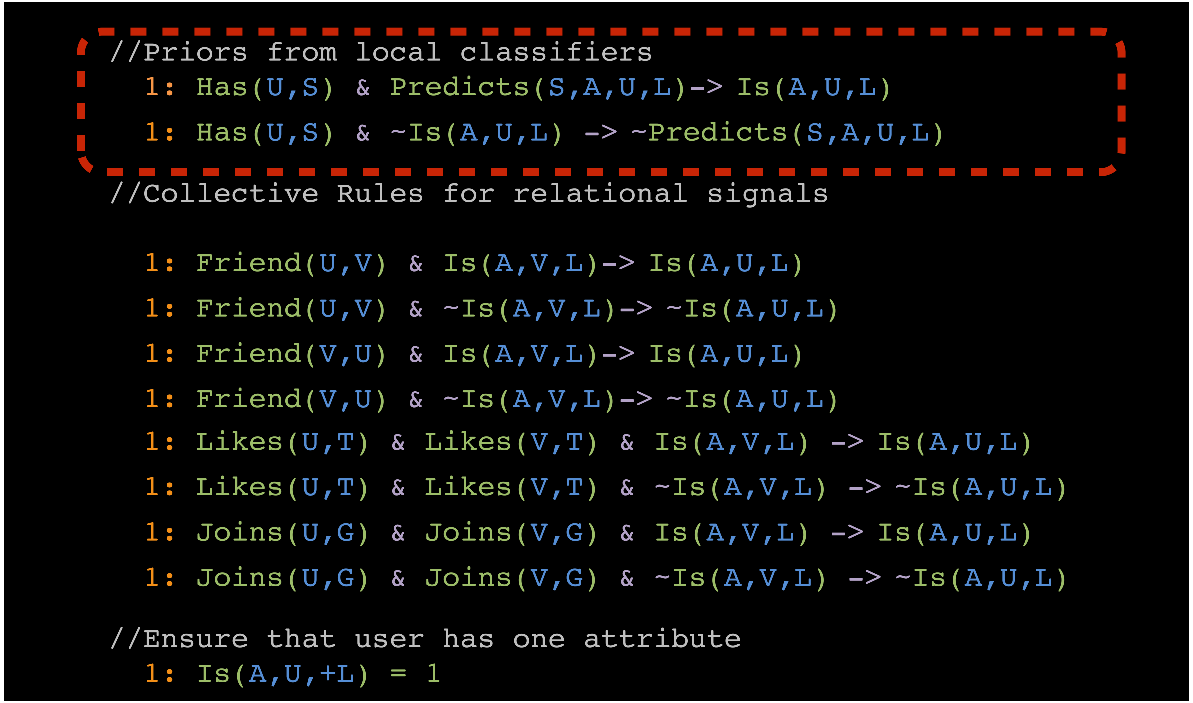


```
//Priors from local classifiers
  1: Has(U,S) & Predicts(S,A,U,L)-> Is(A,U,L)
  1: Has(U,S) & ~Is(A,U,L) -> ~Predicts(S,A,U,L)

//Collective Rules for relational signals

  1: Friend(U,V) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(U,V) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Friend(V,U) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(V,U) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & Is(A,V,L) -> Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & ~Is(A,V,L) -> ~Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & Is(A,V,L) -> Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & ~Is(A,V,L) -> ~Is(A,U,L)

//Ensure that user has one attribute
  1: Is(A,U,+L) = 1
```

Data file for User Modeling

```
predicates:

    Predicts/4: closed
    Friend/2: closed
    Likes/2: closed
    Joins/2: closed
    Has/2: closed
    Is/3: open
observations:

    Predicts: ../data/local_predictor_obs.txt
    Has: ../data/has_obs.txt
    Friend: ../data/friend_obs.txt
    Likes : ../data/likes_obs.txt
    Joins : ../data/joins_obs.txt
    Is : ../data/user_train.txt
targets:

    Is : ../data/user_target.txt
truth:

    Is : ../data/user_truth.txt
```

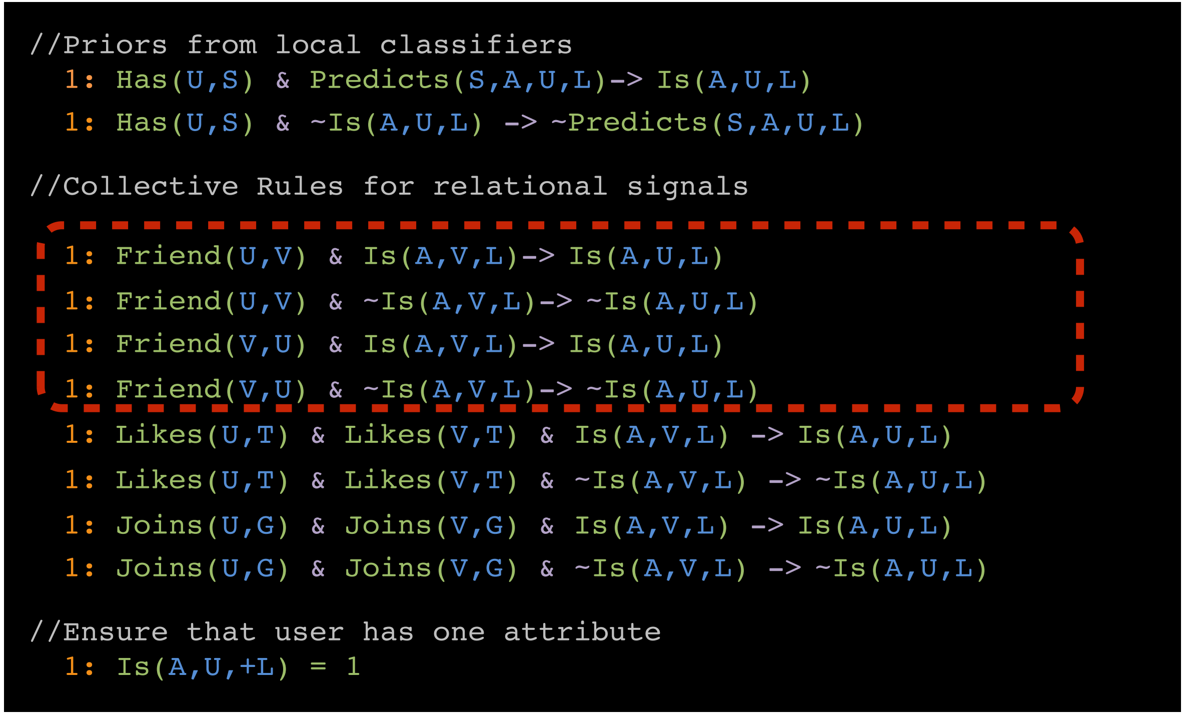
PSL Model for User Modeling



```
//Priors from local classifiers
  1: Has(U,S) & Predicts(S,A,U,L)-> Is(A,U,L)
  1: Has(U,S) & ~Is(A,U,L) -> ~Predicts(S,A,U,L)
//Collective Rules for relational signals

  1: Friend(U,V) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(U,V) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Friend(V,U) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(V,U) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & Is(A,V,L) -> Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & ~Is(A,V,L) -> ~Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & Is(A,V,L) -> Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & ~Is(A,V,L) -> ~Is(A,U,L)

//Ensure that user has one attribute
  1: Is(A,U,+L) = 1
```

local predictor

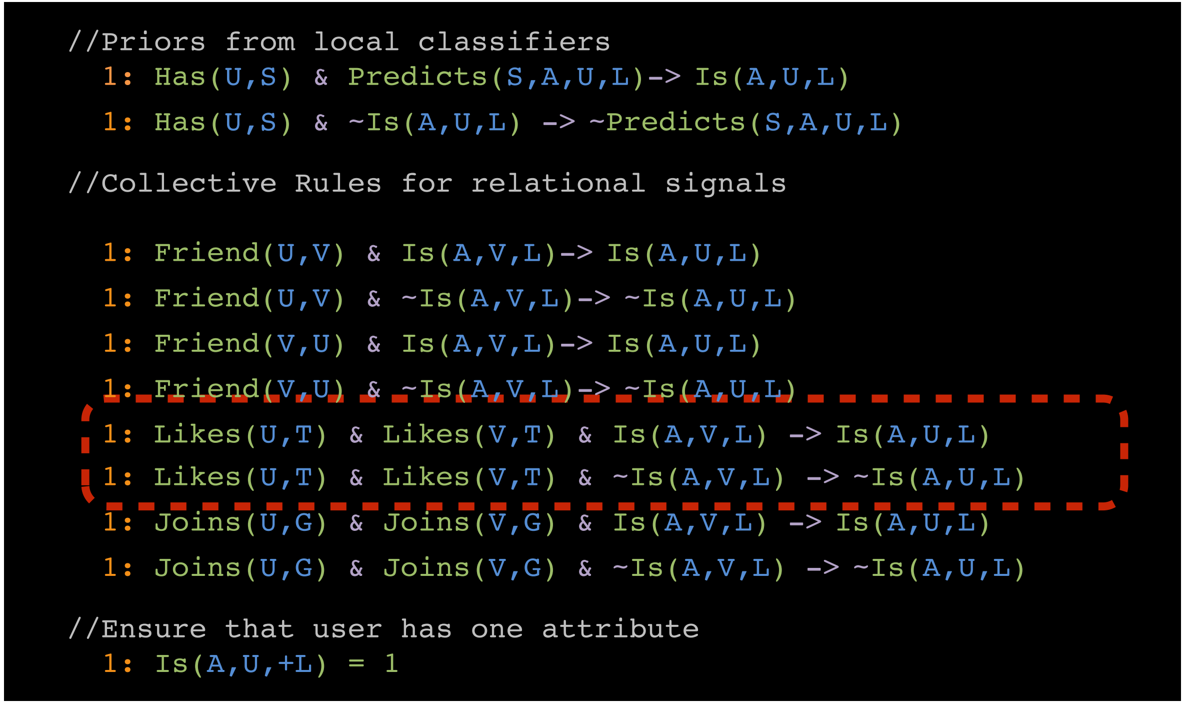
PSL Model for User Modeling



```
//Priors from local classifiers
  1: Has(U,S) & Predicts(S,A,U,L)-> Is(A,U,L)
  1: Has(U,S) & ~Is(A,U,L) -> ~Predicts(S,A,U,L)

//Collective Rules for relational signals
  1: Friend(U,V) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(U,V) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Friend(V,U) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(V,U) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & Is(A,V,L) -> Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & ~Is(A,V,L) -> ~Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & Is(A,V,L) -> Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & ~Is(A,V,L) -> ~Is(A,U,L)

//Ensure that user has one attribute
  1: Is(A,U,+L) = 1
```

Friend

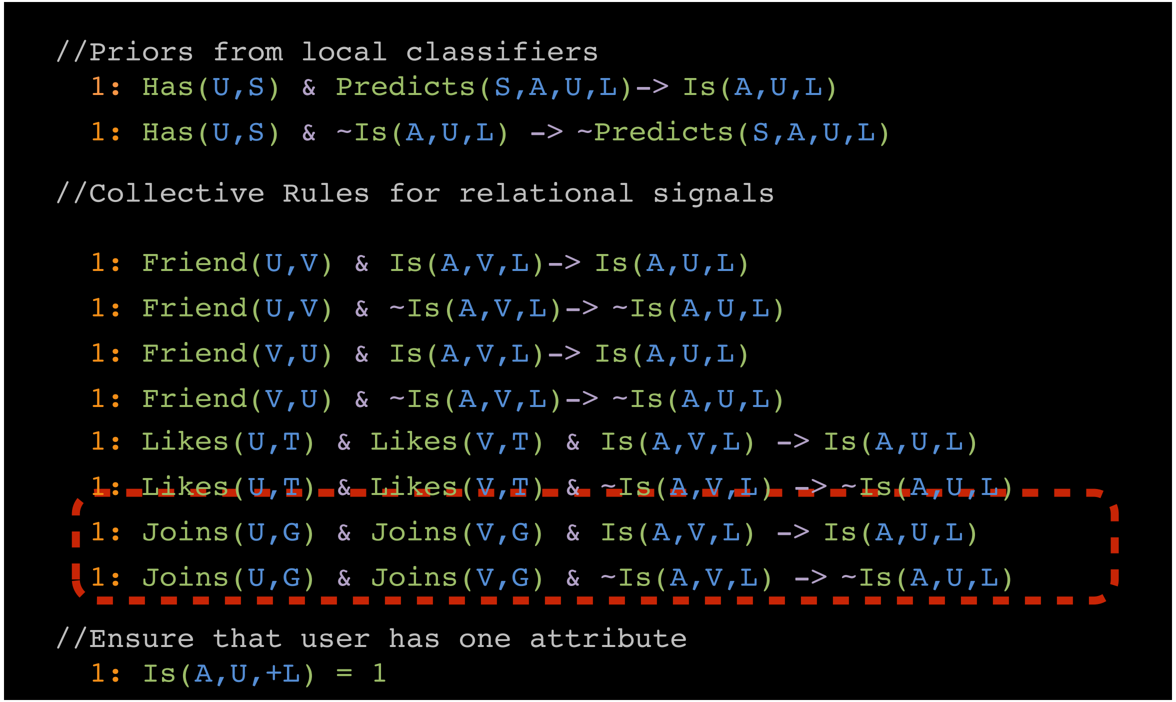
PSL Model for User Modeling



```
//Priors from local classifiers
  1: Has(U,S) & Predicts(S,A,U,L)-> Is(A,U,L)
  1: Has(U,S) & ~Is(A,U,L) -> ~Predicts(S,A,U,L)

//Collective Rules for relational signals

  1: Friend(U,V) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(U,V) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Friend(V,U) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(V,U) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & Is(A,V,L) -> Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & ~Is(A,V,L) -> ~Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & Is(A,V,L) -> Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & ~Is(A,V,L) -> ~Is(A,U,L)

//Ensure that user has one attribute
  1: Is(A,U,+L) = 1
```

Likes

PSL Model for User Modeling

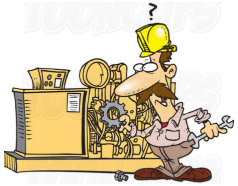


```
//Priors from local classifiers
  1: Has(U,S) & Predicts(S,A,U,L)-> Is(A,U,L)
  1: Has(U,S) & ~Is(A,U,L) -> ~Predicts(S,A,U,L)
//Collective Rules for relational signals

  1: Friend(U,V) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(U,V) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Friend(V,U) & Is(A,V,L)-> Is(A,U,L)
  1: Friend(V,U) & ~Is(A,V,L)-> ~Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & Is(A,V,L) -> Is(A,U,L)
  1: Likes(U,T) & Likes(V,T) & ~Is(A,V,L) -> ~Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & Is(A,V,L) -> Is(A,U,L)
  1: Joins(U,G) & Joins(V,G) & ~Is(A,V,L) -> ~Is(A,U,L)
//Ensure that user has one attribute
  1: Is(A,U,+L) = 1
```

Joins

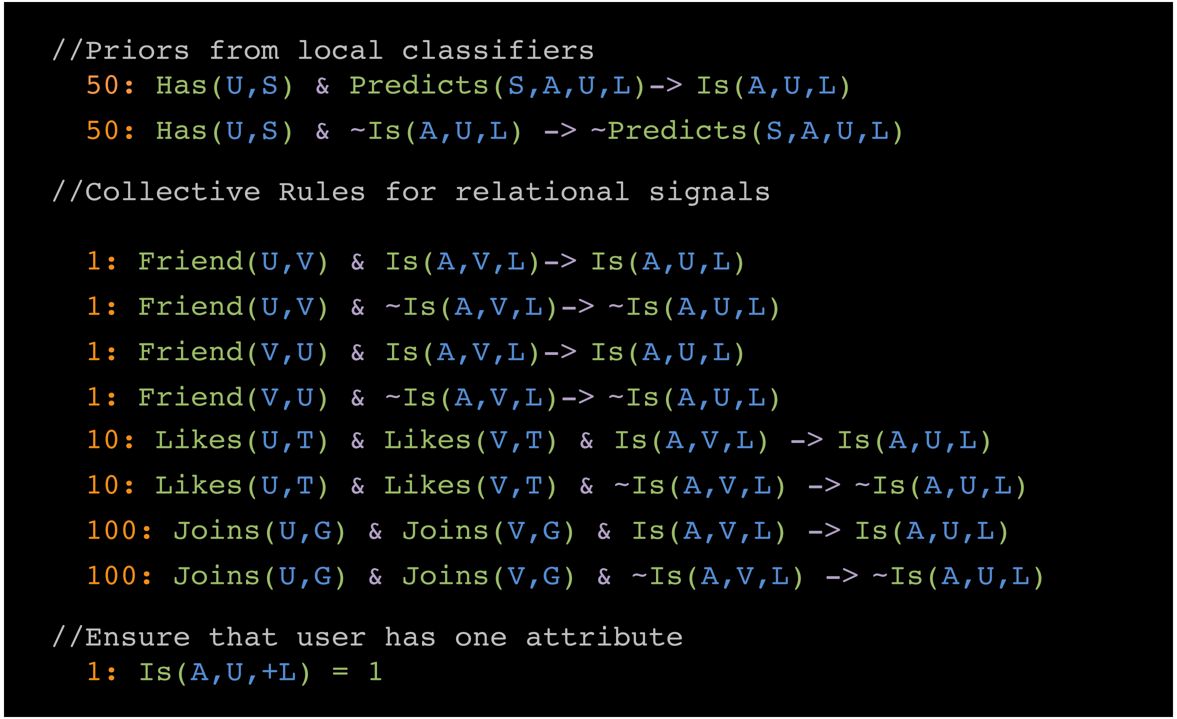
Evaluation Result (Personality prediction-synthetic data)

Type of the model	AUC
Random	0.5
Local Predictor	0.811655
Friendship links	0.528963
Likes	0.724014
Joins	0.865880
All	0.777315



Rules' weights?

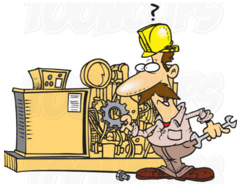
PSL Model for User Modeling



```
//Priors from local classifiers
  50: Has(U,S) & Predicts(S,A,U,L)-> Is(A,U,L)

  50: Has(U,S) & ~Is(A,U,L) -> ~Predicts(S,A,U,L)

//Collective Rules for relational signals

  1: Friend(U,V) & Is(A,V,L)-> Is(A,U,L)

  1: Friend(U,V) & ~Is(A,V,L)-> ~Is(A,U,L)

  1: Friend(V,U) & Is(A,V,L)-> Is(A,U,L)

  1: Friend(V,U) & ~Is(A,V,L)-> ~Is(A,U,L)

  10: Likes(U,T) & Likes(V,T) & Is(A,V,L) -> Is(A,U,L)

  10: Likes(U,T) & Likes(V,T) & ~Is(A,V,L) -> ~Is(A,U,L)

  100: Joins(U,G) & Joins(V,G) & Is(A,V,L) -> Is(A,U,L)

  100: Joins(U,G) & Joins(V,G) & ~Is(A,V,L) -> ~Is(A,U,L)

//Ensure that user has one attribute
  1: Is(A,U,+L) = 1
```

Evaluation Result (Personality prediction-synthetic data)

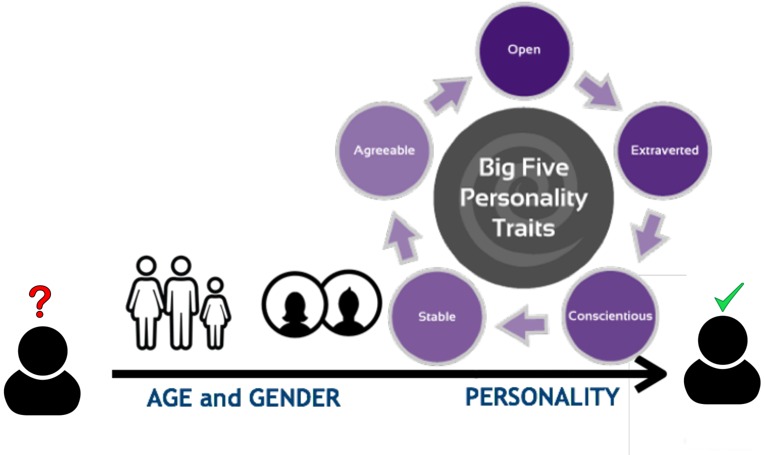
Type of the model	AUC
Random	0.5
Local Predictor	0.811655
Friendship links	0.528963
Likes	0.724014
Joins	0.865880
All	0.777315
All	0.913516



Other combinations?

Weight learning?

Knowledge Fusion Model for User Profiling Based on Multimedia and Multi-Relational User-Generated Content



- Personalised services
- Marketing and advertisement
- Law enforcement
- Employment selection

[Work in progress]

Predicting Users' Age, Gender and Big5 Personality traits

Task: Predicting Facebook Users':

Age, Gender and Big5 Personality traits (Extraversion (Ext), Agreeableness (Agr), Neuroticism (Neu), Openness (Opn), Conscientiousness (Con))

Using **Status updates**, **Profile Picture** and **Facebook Page Likes**

Data: ~6K Facebook users, ~49K Facebook pages and ~725K Page likes Relations

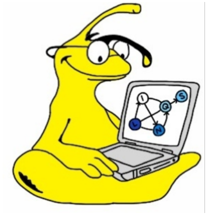
Results: Area under the curve (AUC), 10-fold CV

Model/Characteristic	Gender	Age	Opn	Con	Ext	Agr	Neu
Baseline	0.492	0.488	0.502	0.502	0.506	0.504	0.486
PSL-Textual	0.650	0.710	0.570	0.567	0.553	0.550	0.542
PSL-Visual	0.850	0.579	0.505	0.521	0.531	0.531	0.515
PSL-Relational	0.853	0.881	0.648	0.618	0.592	0.571	0.572
PSL-Fusion	**0.893**	**0.893**	**0.654**	**0.622**	**0.599**	**0.581**	**0.58**

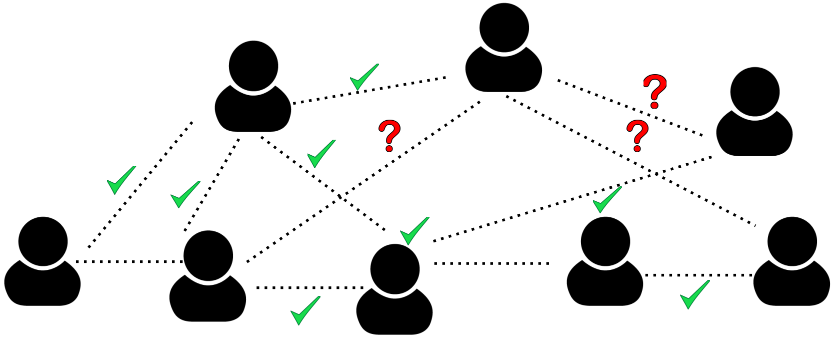
Link Prediction

Eriq Augustine and Golnoosh Farnadi
UC Santa Cruz
MLTrain 2018

psl.linqs.org
github.com/linqs/psl



What is Link Prediction?

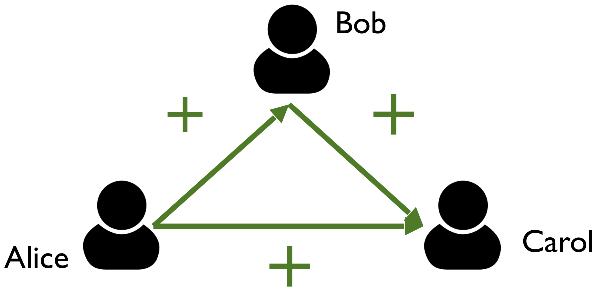


```
Trusts(U,V)
```

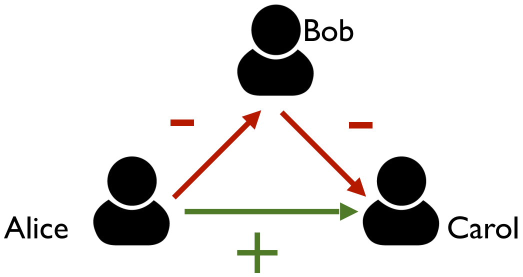
User “**U**” Trust User “**V**”

Social Trust Models: Inferring Trust Networks

- **Model #1: Structural balance** (Granovetter, ’73). Strong ties governed by tendency toward balanced triads. E.g.,
  - ○ “Any friend of yours is a friend of mine.”

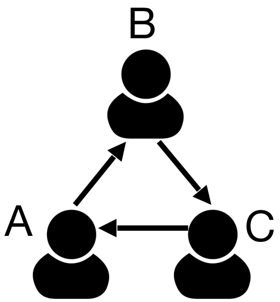


  - ○ “The enemy of my enemy is my friend.”

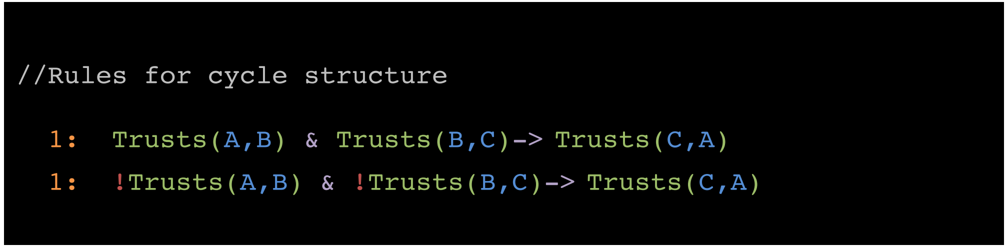


A Flexible Framework for Probabilistic Models of Social Trust, Huang, Kimmig, Getoor, Golbeck, International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction (SBP), 2013

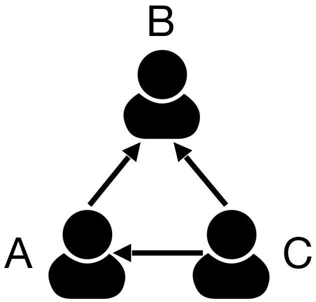
PSL Structural Balance



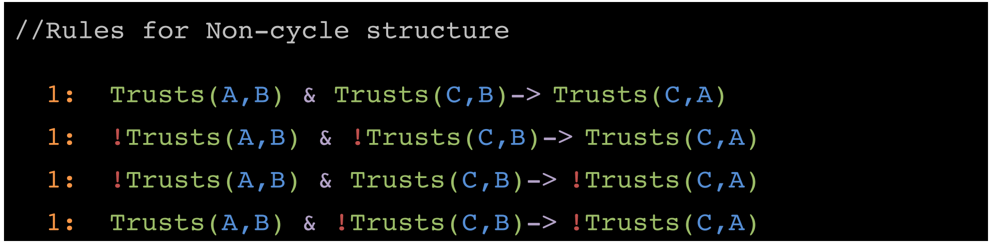
Cycle Structure



```
//Rules for cycle structure

  1:  Trusts(A,B) & Trusts(B,C)-> Trusts(C,A)
  1:  !Trusts(A,B) & !Trusts(B,C)-> Trusts(C,A)
```



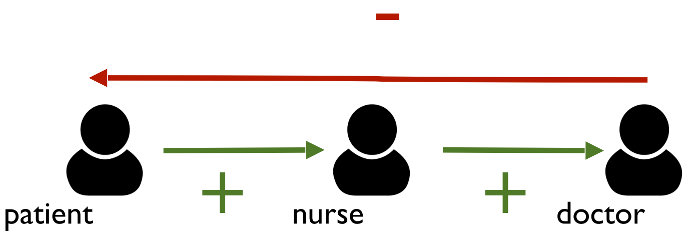
Non-Cycle Structure



```
//Rules for Non-cycle structure

  1:  Trusts(A,B) & Trusts(C,B)-> Trusts(C,A)
  1:  !Trusts(A,B) & !Trusts(C,B)-> Trusts(C,A)
  1:  !Trusts(A,B) & Trusts(C,B)-> !Trusts(C,A)
  1:  Trusts(A,B) & !Trusts(C,B)-> !Trusts(C,A)
```

Social Trust Models: Inferring Trust Networks

- **Model #2: Social status** (Cosmides & Tooby, '92). Strong ties indicate unidirectional respect, "looking up to," expertise status

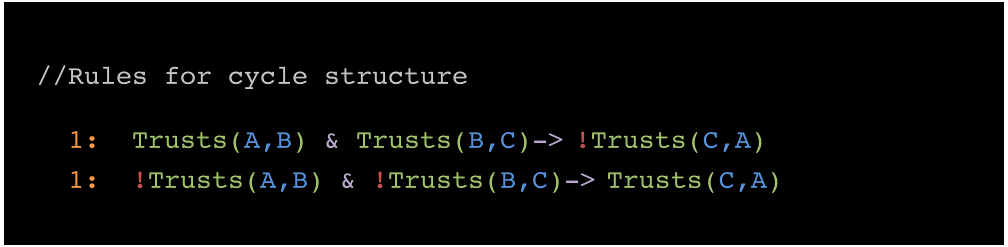


e.g., advisor-advisee, patient-nurse-doctor

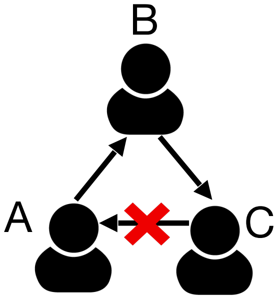
- Leskovec et al. (2010) explored occurrence of both models in data and single-edge prediction

PSL Social Status

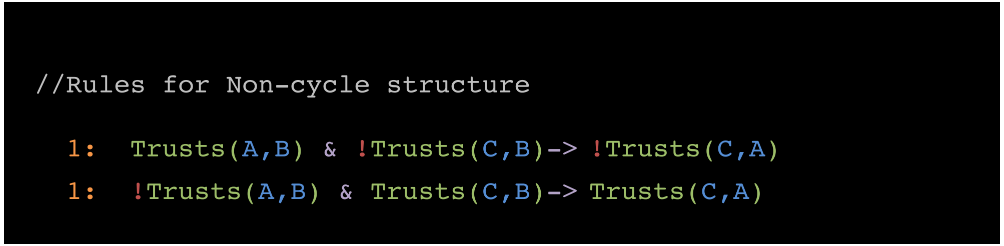
Cycle Structure



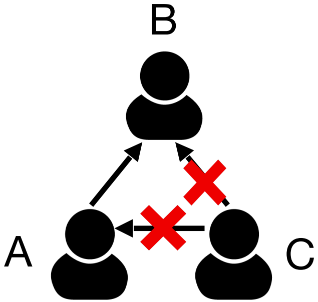
```
//Rules for cycle structure

  1:  Trusts(A,B) & Trusts(B,C)-> !Trusts(C,A)
  1:  !Trusts(A,B) & !Trusts(B,C)-> Trusts(C,A)
```



Non-Cycle Structure

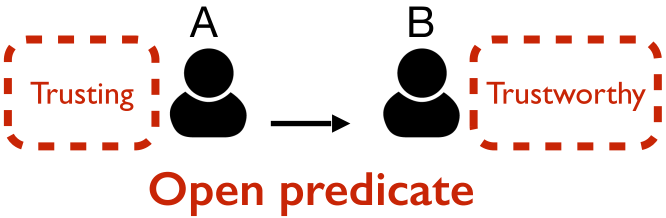


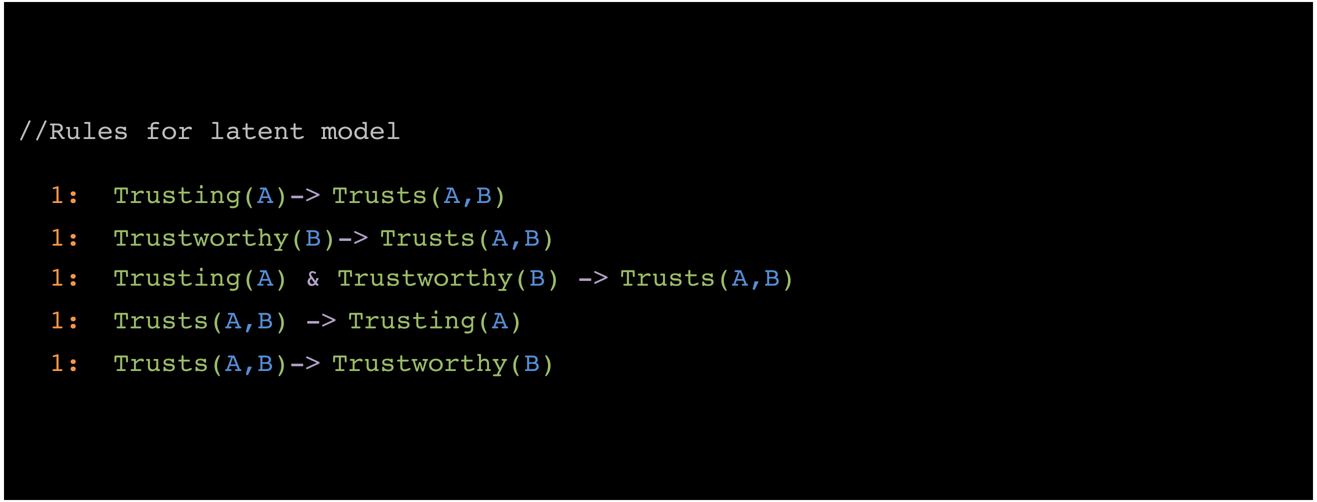
```
//Rules for Non-cycle structure

  1:  Trusts(A,B) & !Trusts(C,B)-> !Trusts(C,A)
  1:  !Trusts(A,B) & Trusts(C,B)-> Trusts(C,A)
```



Latent Variable Model

- **Model #3: Latent model**





```
//Rules for latent model

  1:   Trusting(A)-> Trusts(A,B)
  1:   Trustworthy(B)-> Trusts(A,B)
  1:   Trusting(A) & Trustworthy(B) -> Trusts(A,B)
  1:   Trusts(A,B) -> Trusting(A)
  1:   Trusts(A,B)-> Trustworthy(B)
```

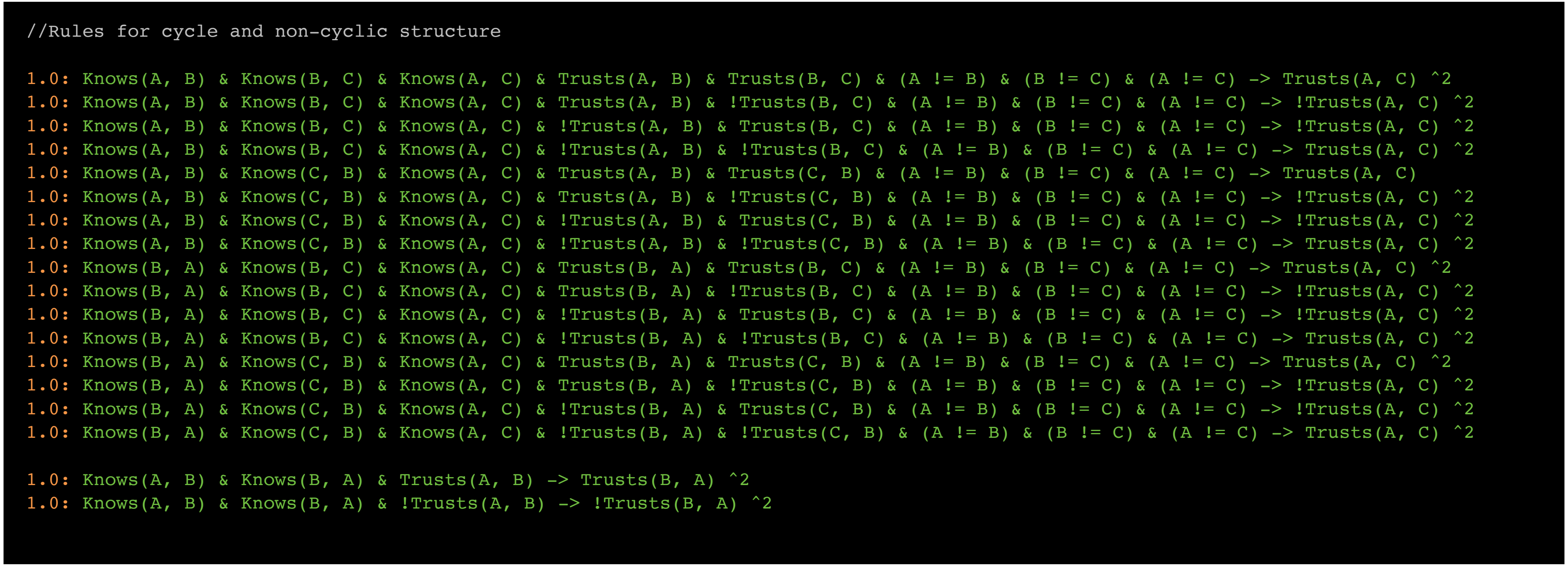
Hands on

- **Data**: 2K user sample of Epinions network and 8.7K signed trust relationships

- git clone https://github.com/linqs/psl-examples.git
- cd psl-examples/trust-prediction/cli
- git checkout uai18

- Models:
  - Balance
  - Status
  - Latent

PSL Model for Trust Prediction (Balance Theory)

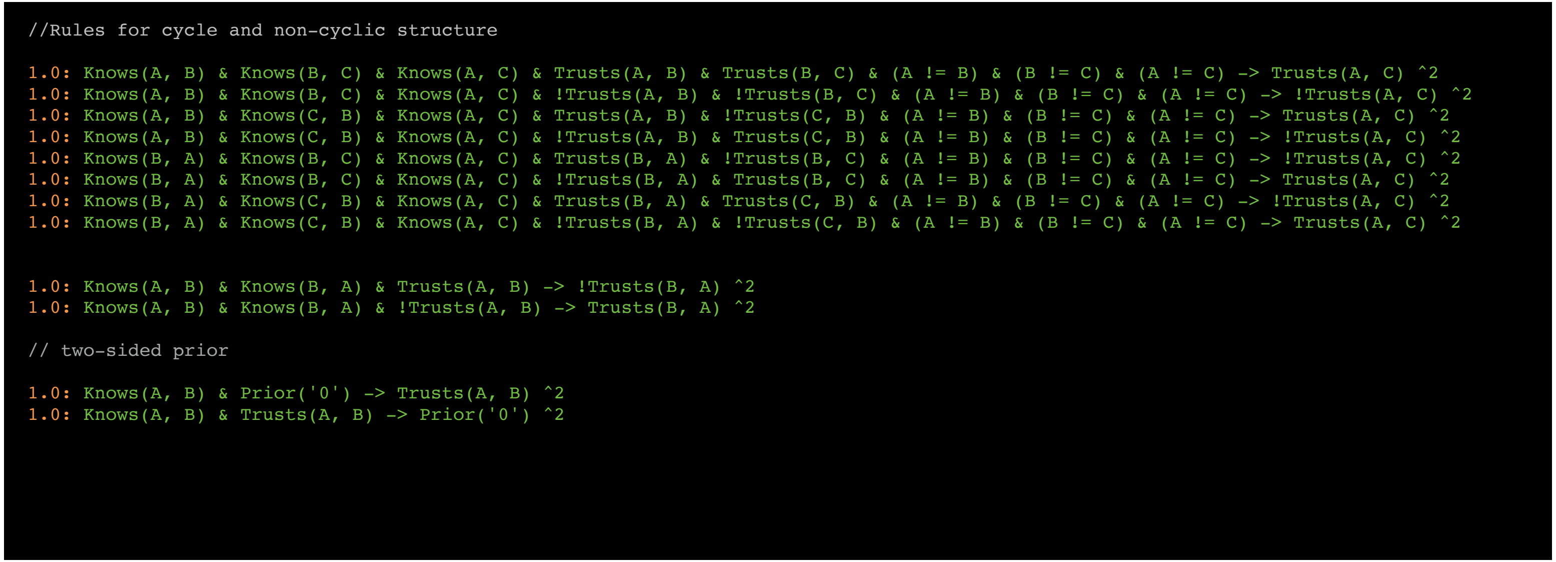


```
//Rules for cycle and non-cyclic structure

1.0: Knows(A, B) & Knows(B, C) & Knows(A, C) & Trusts(A, B) & Trusts(B, C) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(B, C) & Knows(A, C) & Trusts(A, B) & !Trusts(B, C) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(B, C) & Knows(A, C) & !Trusts(A, B) & Trusts(B, C) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(B, C) & Knows(A, C) & !Trusts(A, B) & !Trusts(B, C) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(C, B) & Knows(A, C) & Trusts(A, B) & Trusts(C, B) & (A != B) & (B != C) & (A != C) -> Trusts(A, C)
1.0: Knows(A, B) & Knows(C, B) & Knows(A, C) & Trusts(A, B) & !Trusts(C, B) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(C, B) & Knows(A, C) & !Trusts(A, B) & Trusts(C, B) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(C, B) & Knows(A, C) & !Trusts(A, B) & !Trusts(C, B) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(B, C) & Knows(A, C) & Trusts(B, A) & Trusts(B, C) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(B, C) & Knows(A, C) & Trusts(B, A) & !Trusts(B, C) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(B, C) & Knows(A, C) & !Trusts(B, A) & Trusts(B, C) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(B, C) & Knows(A, C) & !Trusts(B, A) & !Trusts(B, C) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(C, B) & Knows(A, C) & Trusts(B, A) & Trusts(C, B) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(C, B) & Knows(A, C) & Trusts(B, A) & !Trusts(C, B) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(C, B) & Knows(A, C) & !Trusts(B, A) & Trusts(C, B) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(C, B) & Knows(A, C) & !Trusts(B, A) & !Trusts(C, B) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2

1.0: Knows(A, B) & Knows(B, A) & Trusts(A, B) -> Trusts(B, A) ^2
1.0: Knows(A, B) & Knows(B, A) & !Trusts(A, B) -> !Trusts(B, A) ^2
```

Data file for Trust prediction

```
predicates:
    Trusts/2: open
    Knows/2: closed
    Prior/1: closed

observations:
    Trusts: ../data/trust-prediction/eval/trusts_obs.txt
    Knows: ../data/trust-prediction/eval/knows_obs.txt
    Prior: ../data/trust-prediction/eval/prior_obs.txt

targets:
    Trusts: ../data/trust-prediction/eval/trusts_target.txt

truth:
    Trusts: ../data/trust-prediction/eval/trusts_truth.txt
```

PSL Model for Trust Prediction (Social Status)

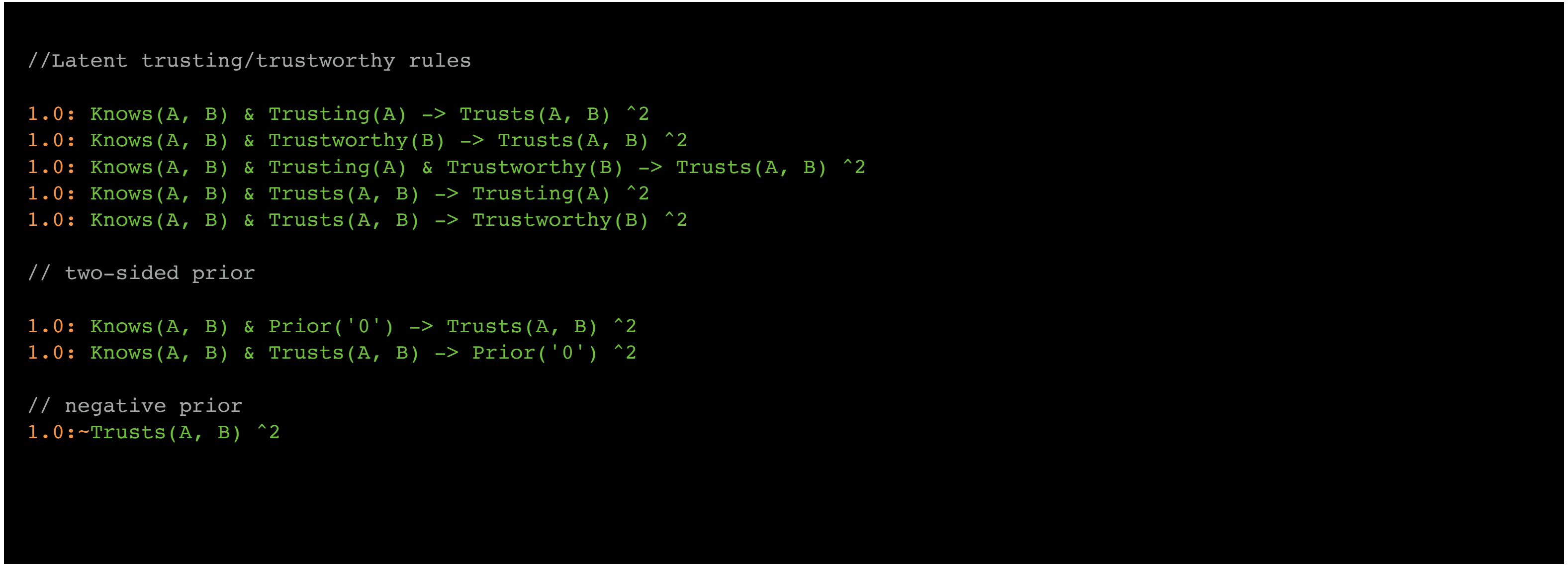


```
//Rules for cycle and non-cyclic structure

1.0: Knows(A, B) & Knows(B, C) & Knows(A, C) & Trusts(A, B) & Trusts(B, C) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(B, C) & Knows(A, C) & !Trusts(A, B) & !Trusts(B, C) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(C, B) & Knows(A, C) & Trusts(A, B) & !Trusts(C, B) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(A, B) & Knows(C, B) & Knows(A, C) & !Trusts(A, B) & Trusts(C, B) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(B, C) & Knows(A, C) & Trusts(B, A) & !Trusts(B, C) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(B, C) & Knows(A, C) & !Trusts(B, A) & Trusts(B, C) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(C, B) & Knows(A, C) & Trusts(B, A) & Trusts(C, B) & (A != B) & (B != C) & (A != C) -> !Trusts(A, C) ^2
1.0: Knows(B, A) & Knows(C, B) & Knows(A, C) & !Trusts(B, A) & !Trusts(C, B) & (A != B) & (B != C) & (A != C) -> Trusts(A, C) ^2


1.0: Knows(A, B) & Knows(B, A) & Trusts(A, B) -> !Trusts(B, A) ^2
1.0: Knows(A, B) & Knows(B, A) & !Trusts(A, B) -> Trusts(B, A) ^2

// two-sided prior

1.0: Knows(A, B) & Prior('0') -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trusts(A, B) -> Prior('0') ^2
```

Get the status model from:

- psl-examples/trust-prediction/cli/alternate-models

PSL Model for Trust Prediction (Latent)



```
//Latent trusting/trustworthy rules

1.0: Knows(A, B) & Trusting(A) -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trustworthy(B) -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trusting(A) & Trustworthy(B) -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trusts(A, B) -> Trusting(A) ^2
1.0: Knows(A, B) & Trusts(A, B) -> Trustworthy(B) ^2

// two-sided prior

1.0: Knows(A, B) & Prior('0') -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trusts(A, B) -> Prior('0') ^2

// negative prior
1.0:~Trusts(A, B) ^2
```

Get the latent model and data from:

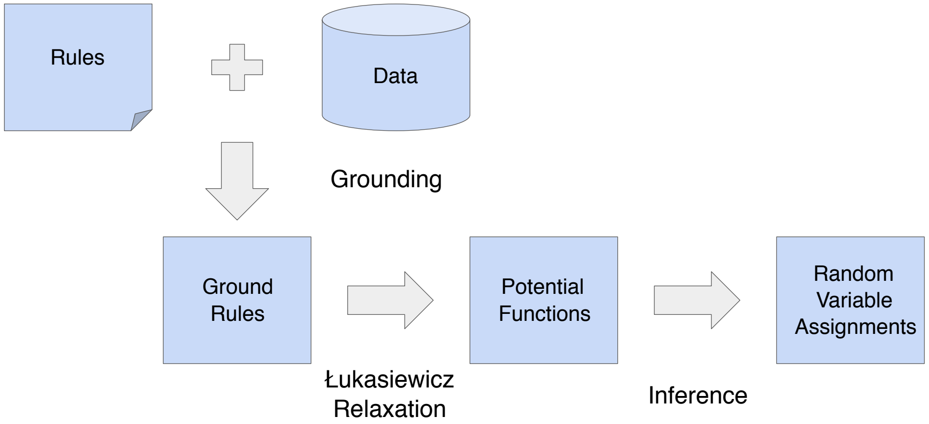
- psl-examples/trust-prediction/cli/alternate-models

Data file for Trust prediction (Latent model)

```
predicates:
    Trusts/2: open
    Trusting/1: open
    Trustworthy/1: open
    Knows/2: closed
    Prior/1: closed

observations:
    Trusts: ../data/trust-prediction/eval/trusts_obs.txt
    Knows: ../data/trust-prediction/eval/knows_obs.txt
    Prior: ../data/trust-prediction/eval/prior_obs.txt

targets:
    Trusts: ../data/trust-prediction/eval/trusts_target.txt

truth:
    Trusts: ../data/trust-prediction/eval/trusts_truth.txt
```

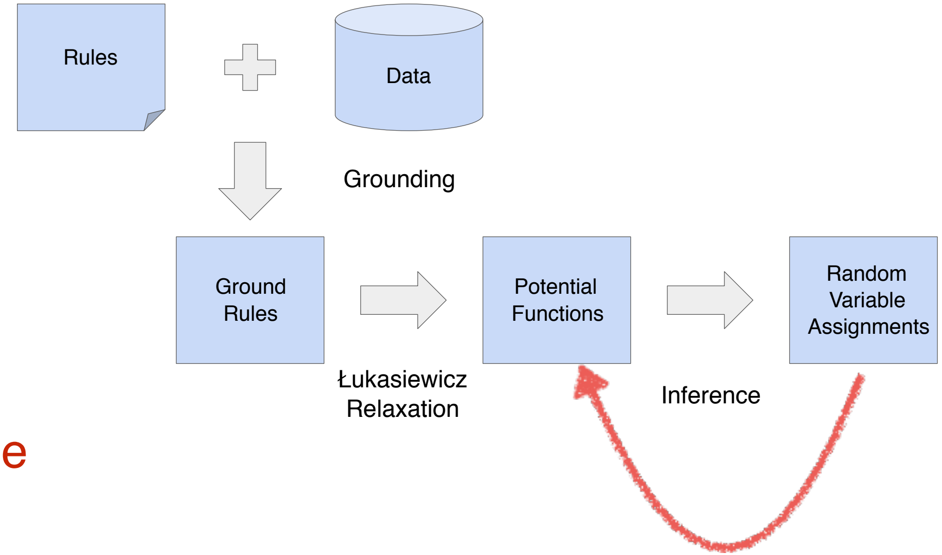
Inference for latent models

MAP Inference



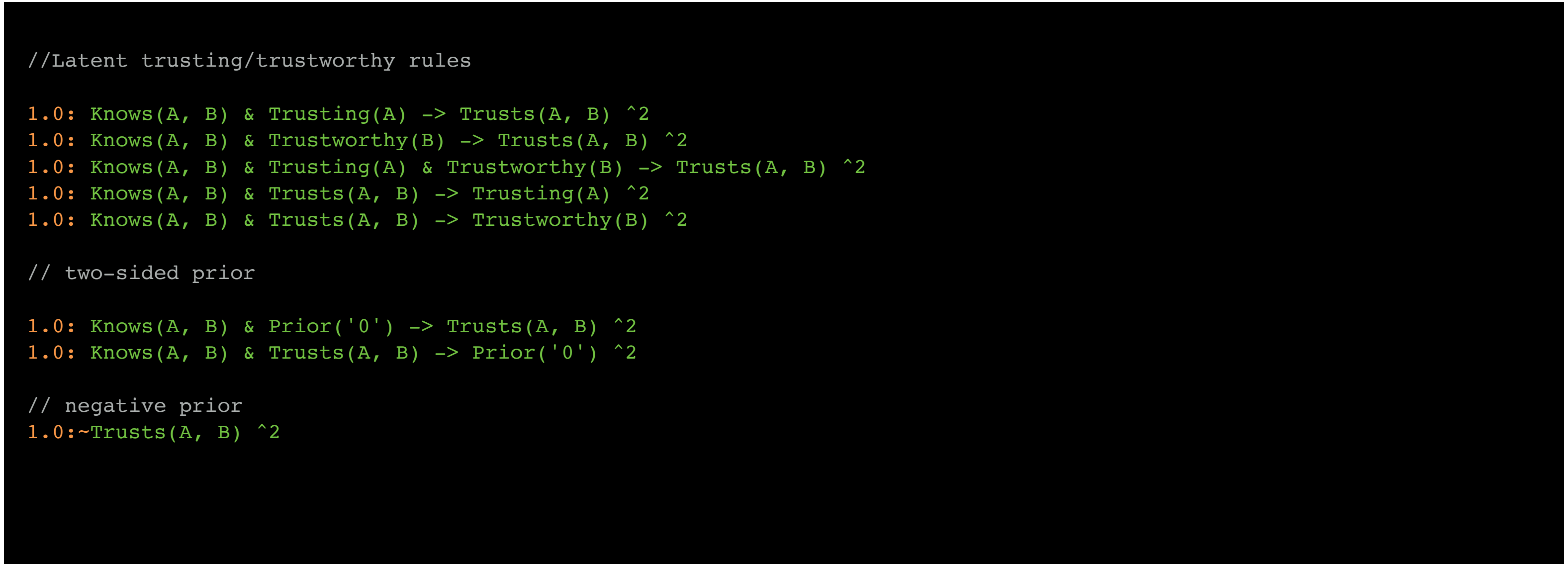
Inference for latent models

MAP Inference



Lazy MAP inference

PSL Model for Trust Prediction (Latent)



```
//Latent trusting/trustworthy rules

1.0: Knows(A, B) & Trusting(A) -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trustworthy(B) -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trusting(A) & Trustworthy(B) -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trusts(A, B) -> Trusting(A) ^2
1.0: Knows(A, B) & Trusts(A, B) -> Trustworthy(B) ^2

// two-sided prior

1.0: Knows(A, B) & Prior('0') -> Trusts(A, B) ^2
1.0: Knows(A, B) & Trusts(A, B) -> Prior('0') ^2

// negative prior
1.0:~Trusts(A, B) ^2
```

Change the parameter in run.sh:
—infer org.linqs.psl.application.inference.LazyMPEInference

Evaluation Result (Trust prediction- Epinions data)

Type of the model	AUC	AUC (positive)	AUC (negative)
Balance Theory	0.808961	0.973752	0.450463
Social Status	0.633428	0.946462	0.231061
Latent Model	0.917668	0.991246	0.557115

Evaluation Result (Trust prediction- Epinions data)

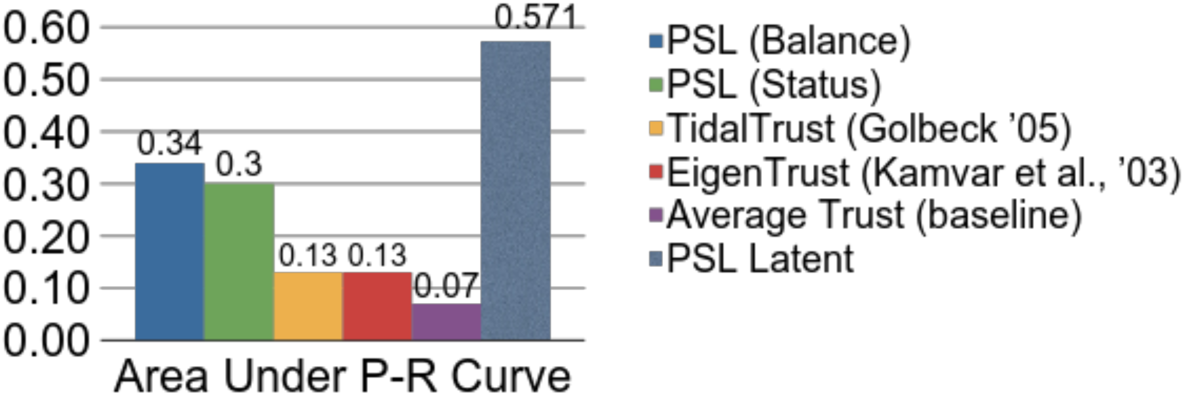
Type of the model	AUC	AUC (positive)	AUC (negative)
Balance Theory	0.808961	0.973752	0.450463
Social Status	0.633428	0.946462	0.231061
Latent Model	0.917668	0.991246	0.557115
?	?	?	?

Weight learning?

Other combinations?

Different rules?

Predicting Distrust



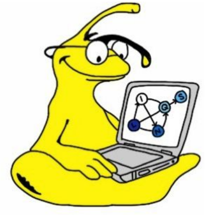
- **Data**: 2K user sample of Epinions network and 8.7K signed trust relationships
- 8-fold cross-validation
- Area under precision-recall curve for rarer **distrust links**

A Flexible Framework for Probabilistic Models of Social Trust, Huang, Kimmig, Getoor, Golbeck,
International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction (SBP), 2013

Entity Resolution

Eriq Augustine and Golnoosh Farnadi
UC Santa Cruz

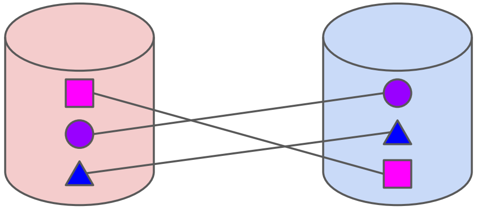
MLTrain 2018



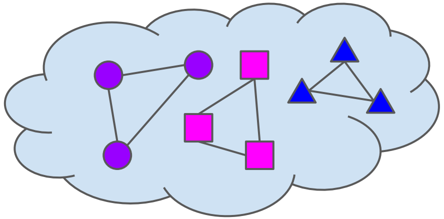
psl.linqs.org
github.com/linqs/psl

What is Entity Resolution?

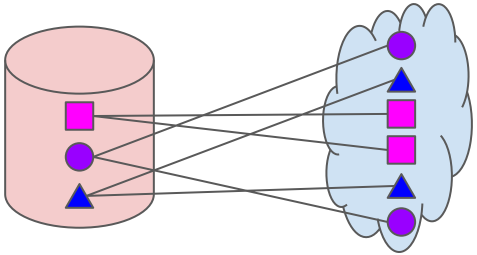
- Entity Resolution comes in several variants:
  - Record Linkage
    - Matching between two (mostly) deduplicated data sources
    - Makes the 1-1 assumption



  - Deduplication
    - Given a single collection of references, find all references that refer to the same entity.



  - Reference Matching
    - Given a deduplicated and a noisy source, match all the noisy references to the deduplicated entities.



Getting the Code

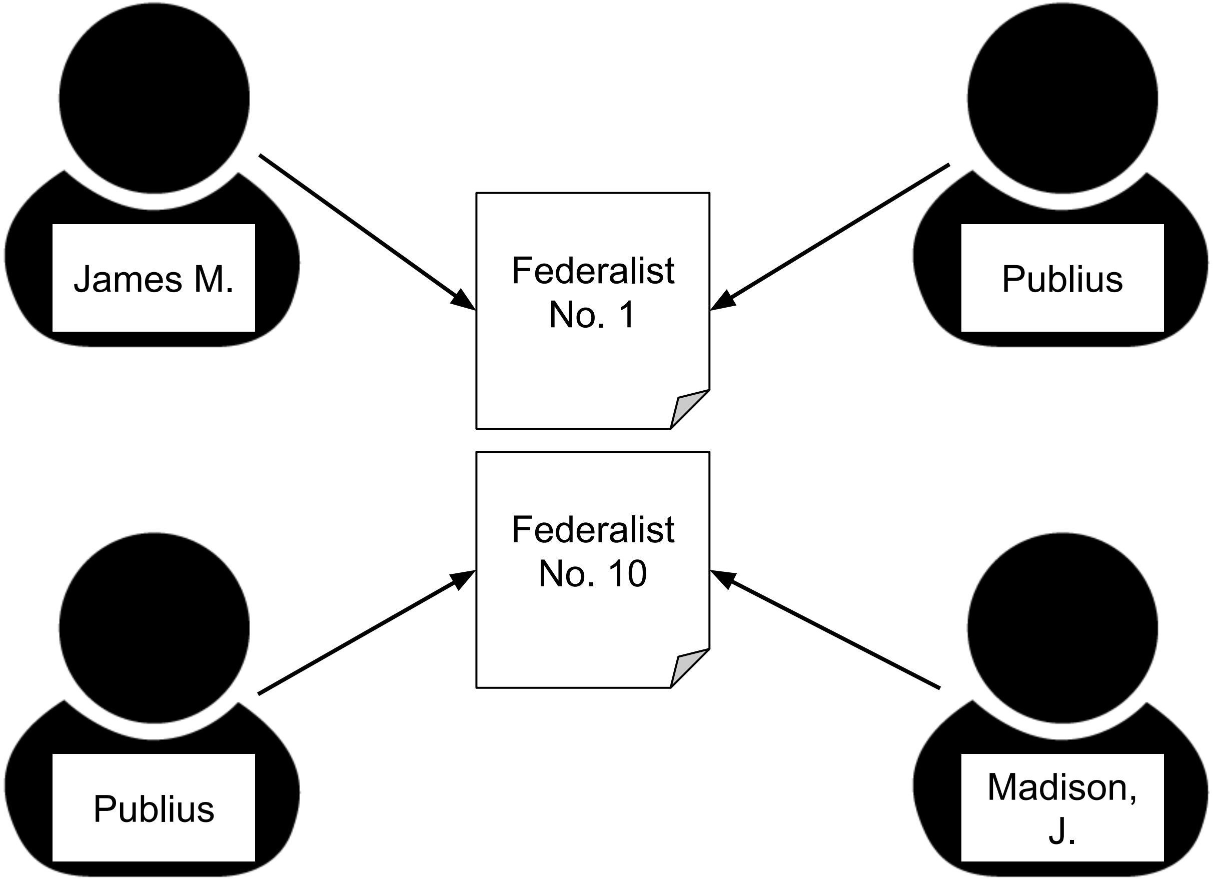
```
git clone https://github.com/linqs/psl-examples.git

cd psl-examples/entity-resolution/cli

git checkout uai18

./run.sh
```

All examples available at: https://github.com/linqs/psl-examples

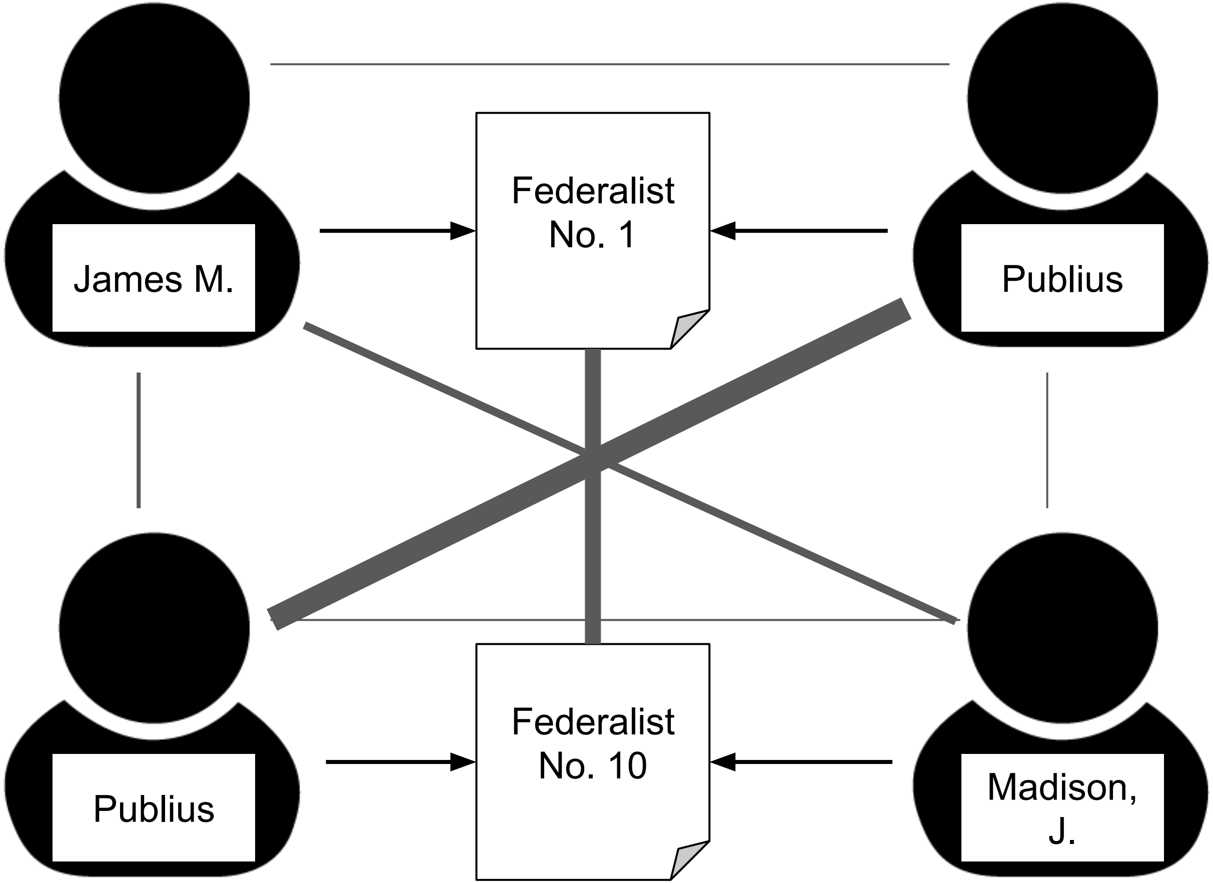
Data

- Citation Network
- Deduplicate
  - Authors
  - Papers
- CiteSeer

Size	Authors	Papers
Small	1136	864
Medium	**1813**	**1143**
Large	2892	1504



Initial Model



James M.

Federalist No. 1

Publius

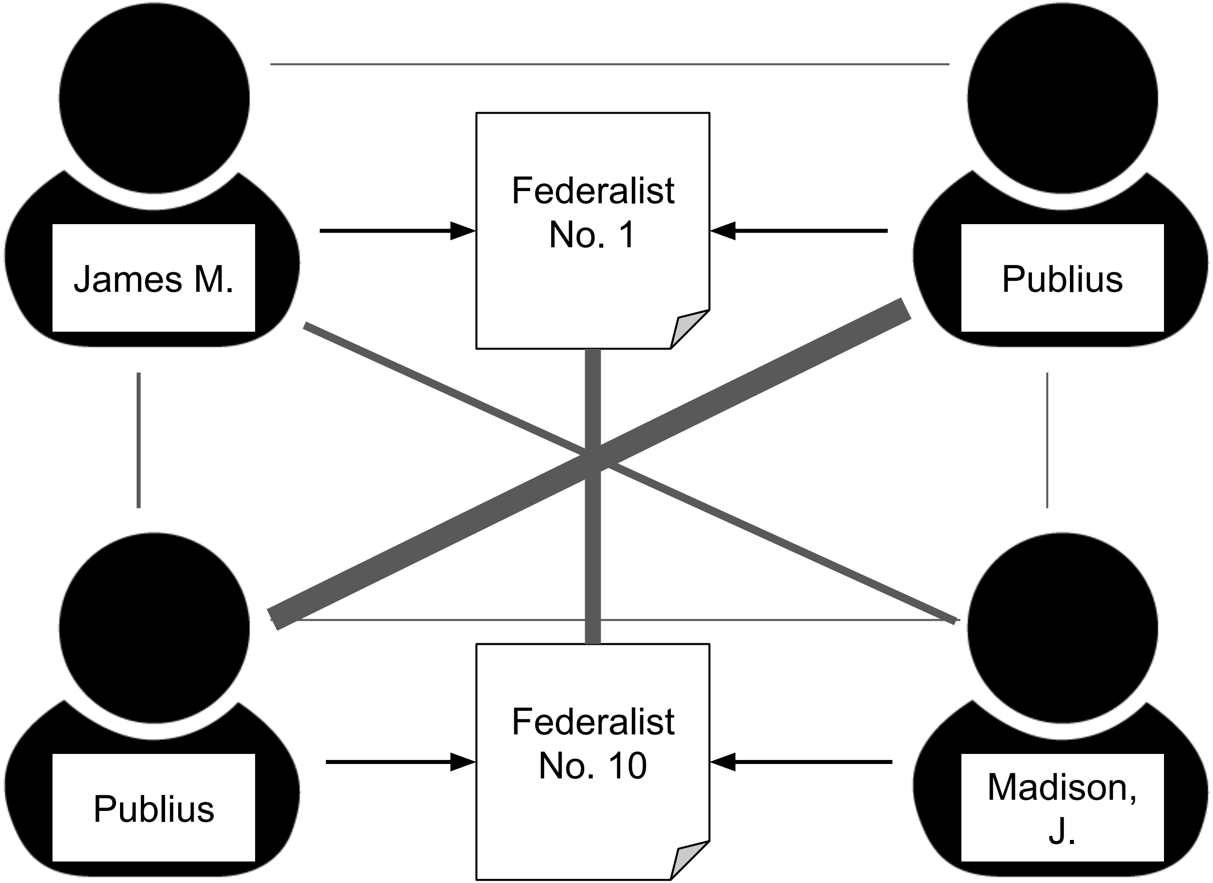
Publius

Federalist No. 10

Madison, J.

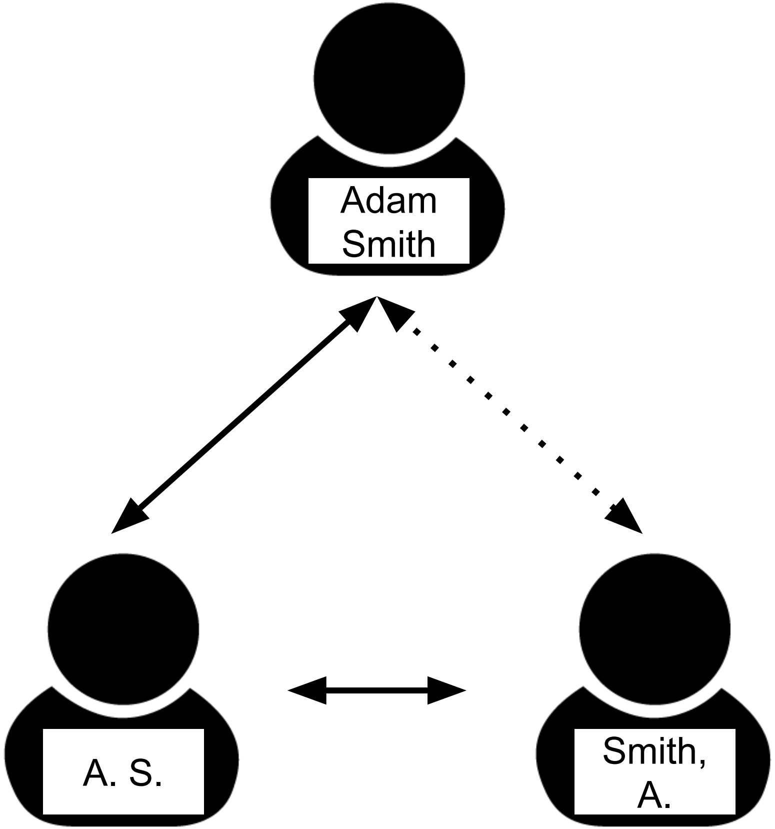
Initial Model

```
AuthorName(A1, N1)
     & AuthorName(A2, N2)
     & SimName(N1, N2)
     -> SameAuthor(A1, A2)


PaperTitle(P1, T1)
     & PaperTitle(P2, T2)
     & SimTitle(T1, T2)
     -> SamePaper(P1, P2)
```



Transitive Equality

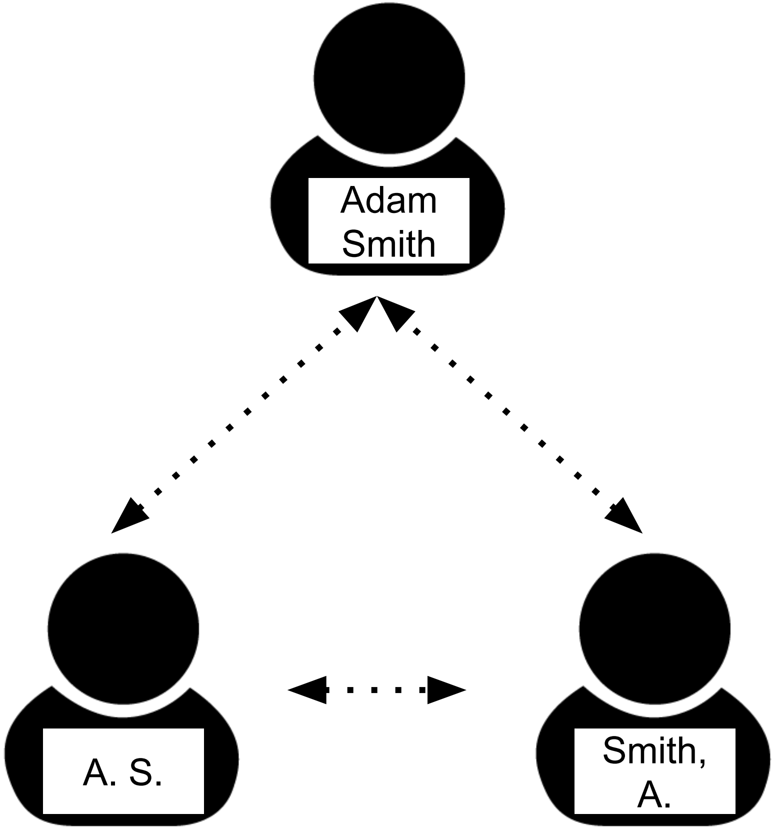
Exploit relational nature of similarity in ER.



Transitive Equality

Exploit relational nature of similarity in ER.

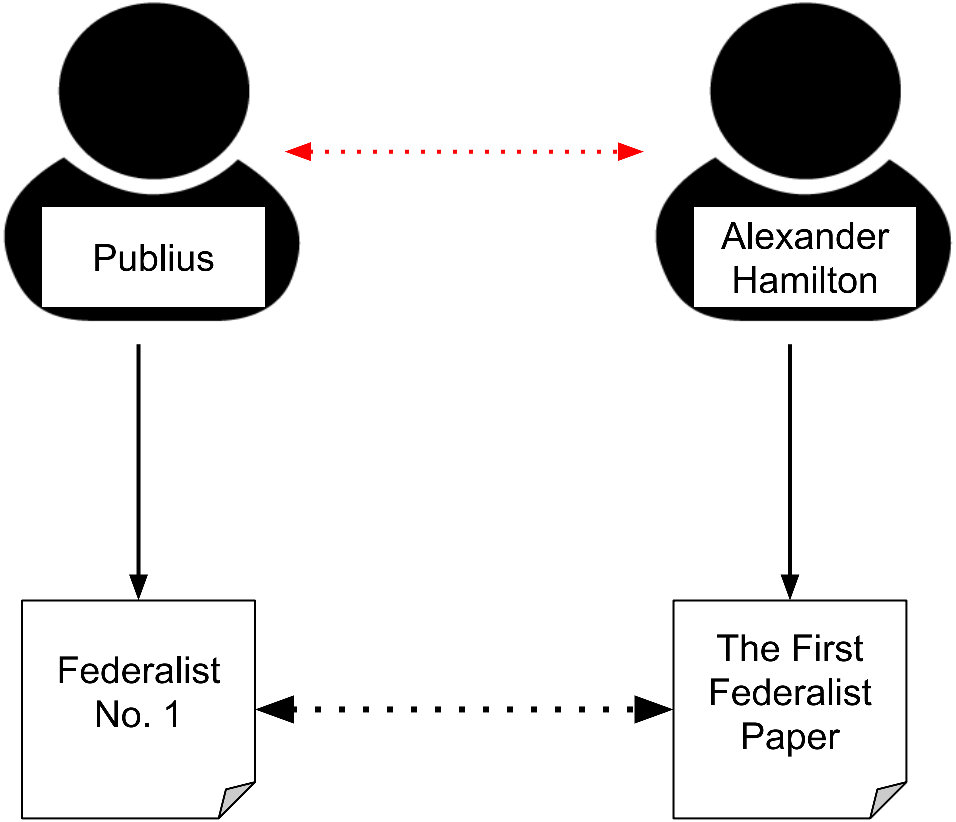
```
SameAuthor(A1, A2)
     & SameAuthor(A2, A3)
     -> SameAuthor(A1, A3)
```



Transitive Relational

What other transitive relational rules can we get?

Transitive Relational



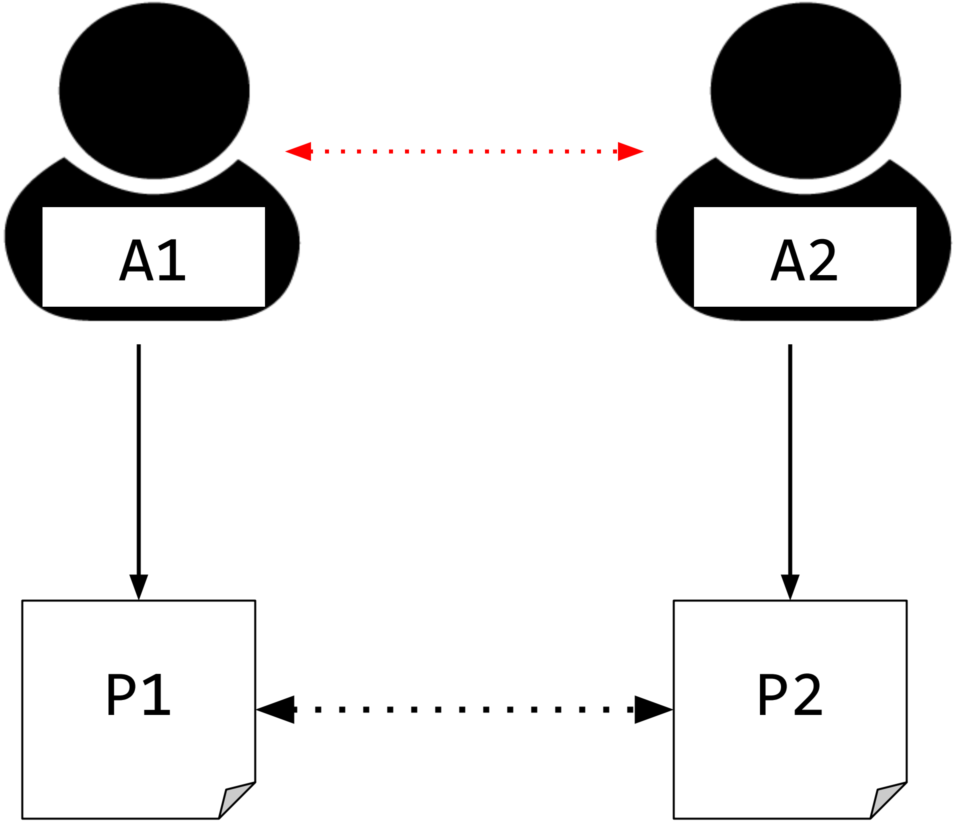
Publius

Alexander Hamilton

Federalist No. 1

The First Federalist Paper

Transitive Relational

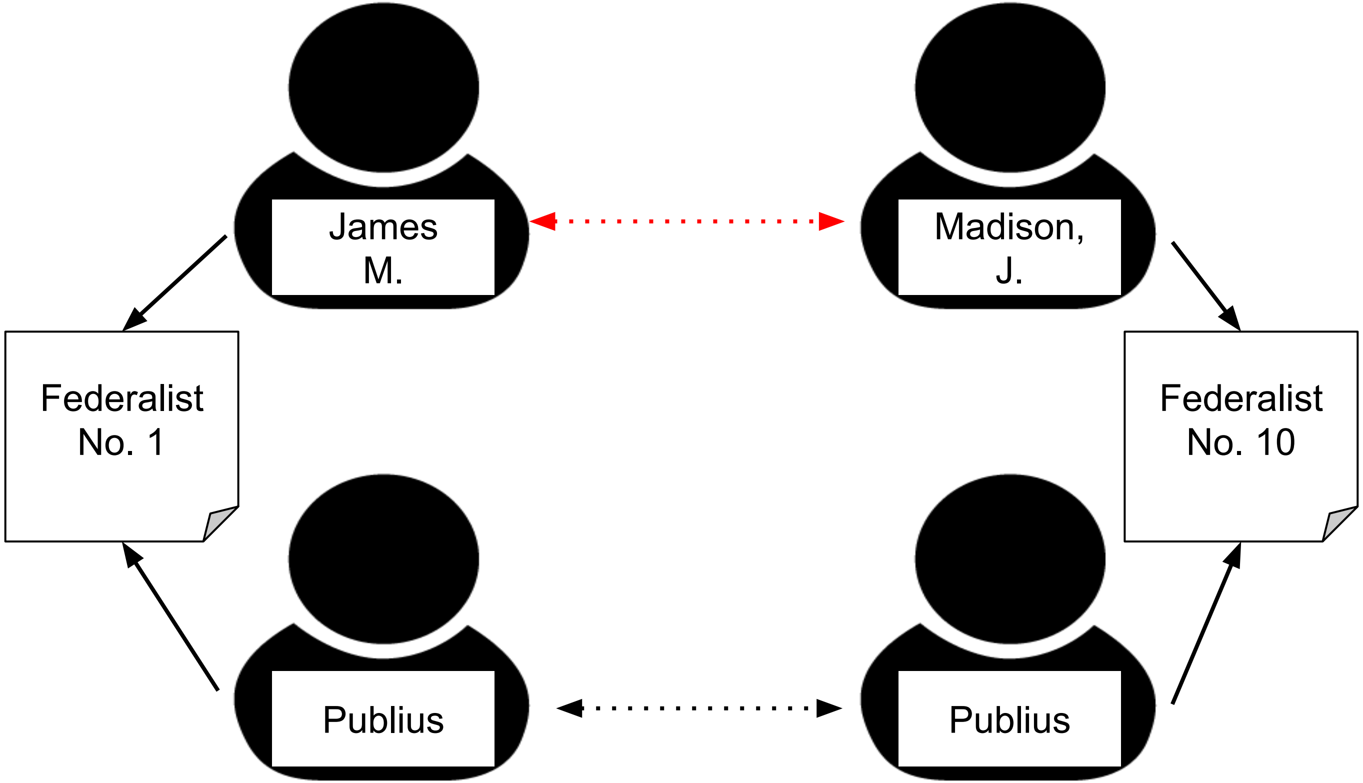


```
AuthorOf(A1, P1) & AuthorOf(A2, P2) & SamePaper(P1, P2) -> SameAuthor(A1, A2)
```

Transitive Relational

What other transitive relational rules can we get?

Transitive Relational



James M.

Madison, J.

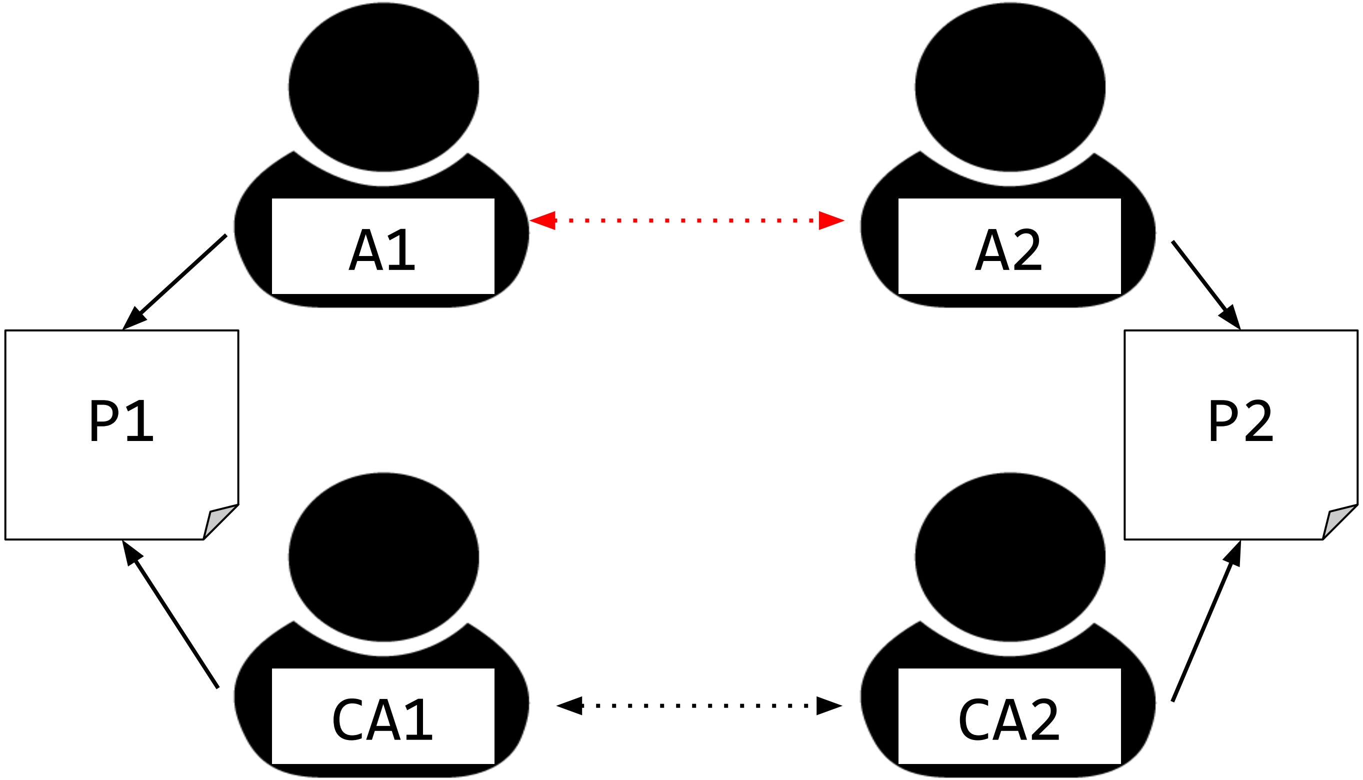
Federalist No. 1

Federalist No. 10

Publius

Publius

Transitive Relational



```
AuthorOf(A1, P1) & AuthorOf(A2, P2) & AuthorOf(CA1, P1) & AuthorOf(CA2, P2)
     & SameAuthor(CA1, CA2) -> SameAuthor(A1, A2)
```

Transitive Blowup!

```
SameAuthor(A1, A2) & SameAuthor(A2, A3) -> SameAuthor(A1, A3)
```

Transitive Blowup!

```
SameAuthor(A1, A2) & SameAuthor(A2, A3) -> SameAuthor(A1, A3)
```

Arbitrarily choose **three** authors.

Recall we have 1813 authors.

Transitive Blowup!

```
SameAuthor(A1, A2) & SameAuthor(A2, A3) -> SameAuthor(A1, A3)
```

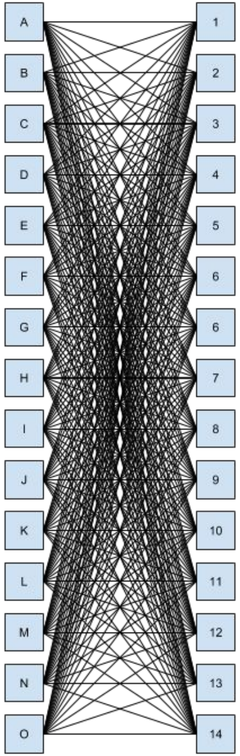
Arbitrarily choose **three** authors.

Recall we have 1813 authors.

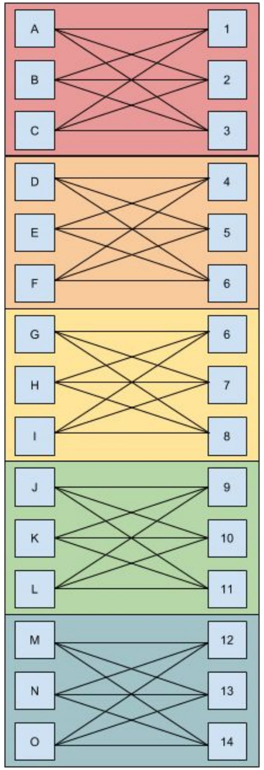
$$\binom{1813}{3}$$

~ 1 Billion Ground Rules

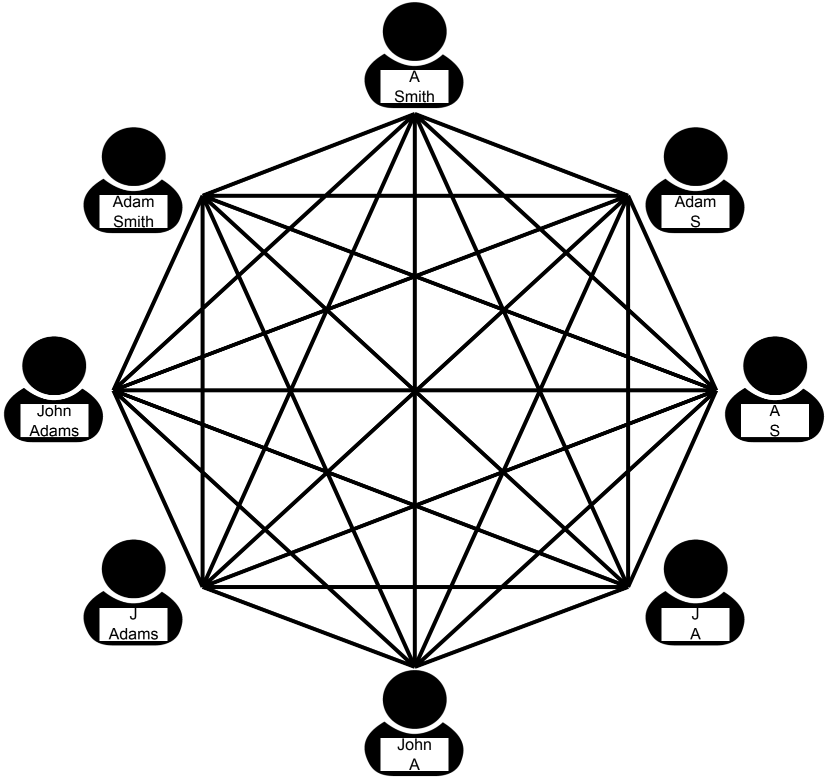
Blocking



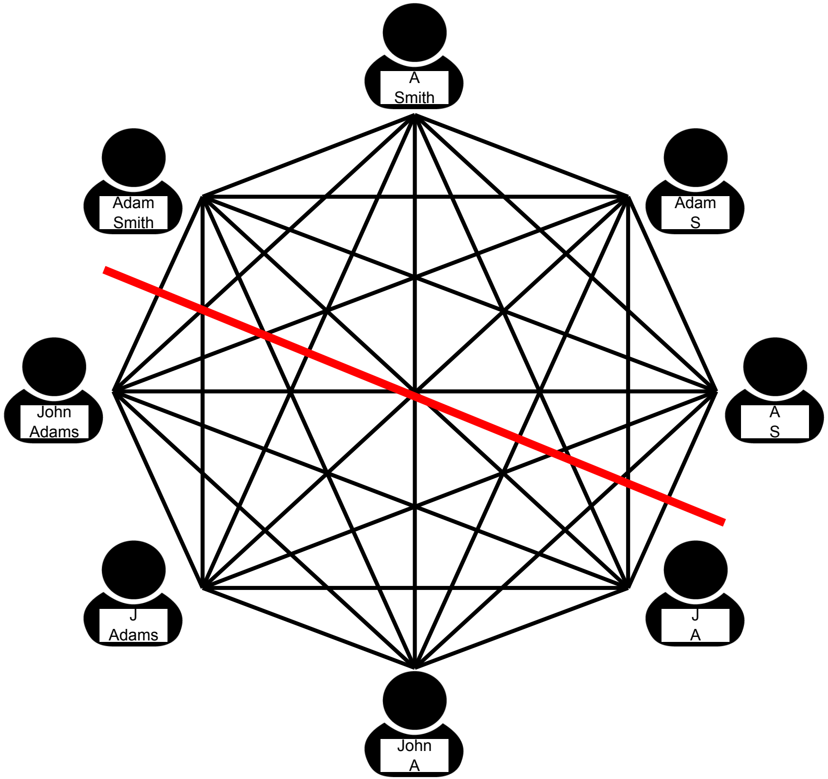
- Blocking is reducing the number of ground potentials using some computed heuristic(s).
- In PSL, this is done by adding predicates that induce sparsity in the MRF.



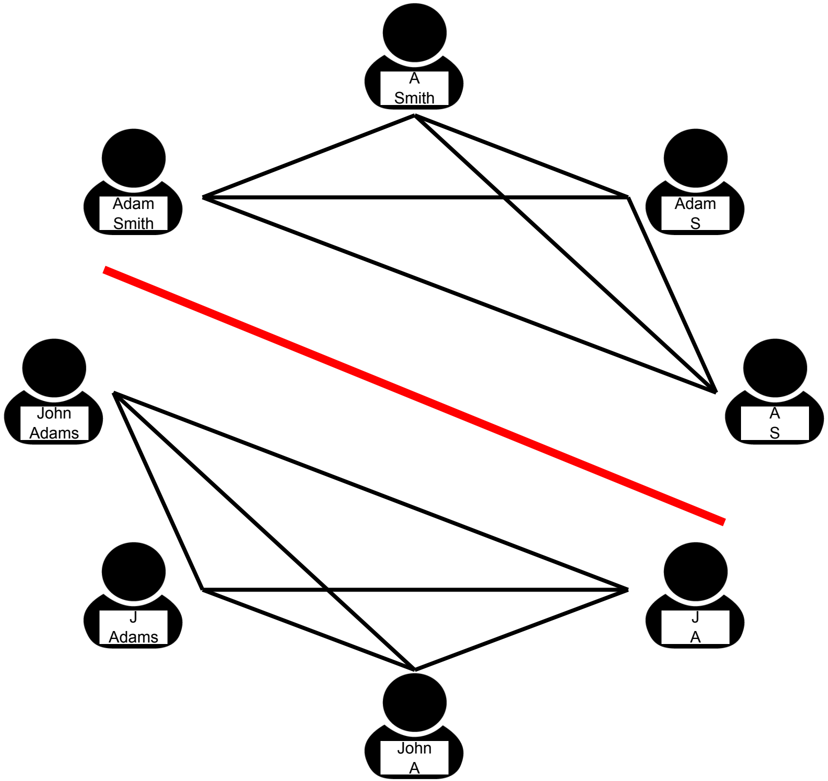
Blocking



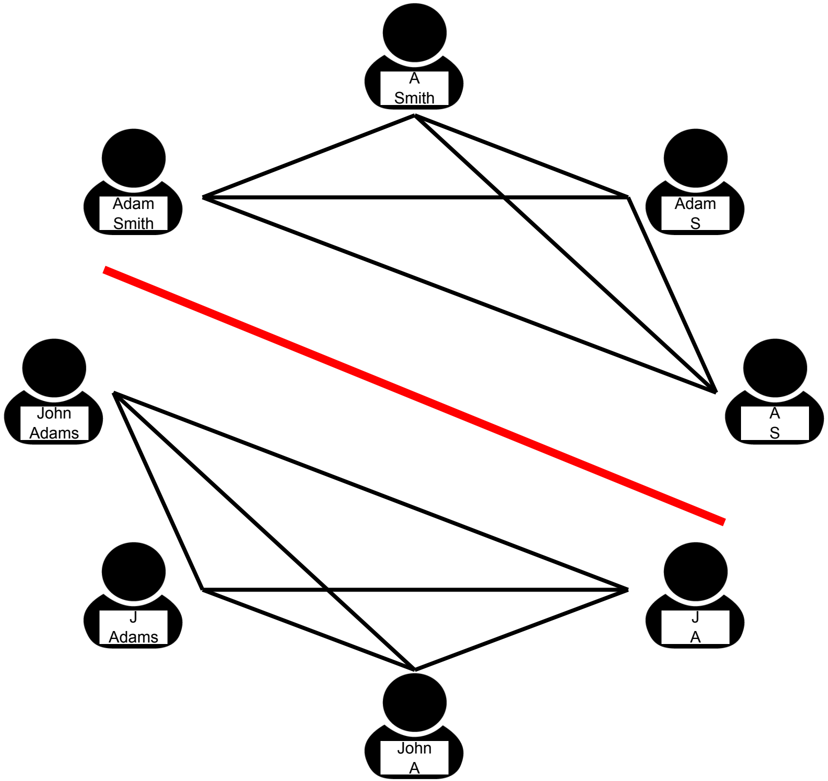
Blocking



Blocking



Blocking



```
AuthorBlock(A1, B) & AuthorBlock(A2, B) & AuthorBlock(A3, B)
    & SameAuthor(A1, A2) & SameAuthor(A2, A3) -> SameAuthor(A1, A3)
```

Blocking

```
AuthorBlock(A1, B) & AuthorBlock(A2, B) & AuthorBlock(A3, B)
     & SameAuthor(A1, A2) & SameAuthor(A2, A3) -> SameAuthor(A1, A3)


AuthorBlock(A1, B1) & AuthorBlock(A2, B1)
     & AuthorBlock(CA1, B2) & AuthorBlock(CA2, B2)
     & AuthorOf(A1, P1) & AuthorOf(A2, P2)
     & AuthorOf(CA1, P1) & AuthorOf(CA2, P2) & SameAuthor(CA1, CA2)
     -> SameAuthor(A1, A2)


AuthorBlock(A1, B) & AuthorBlock(A2, B)
     & AuthorOf(A1, P1) & AuthorOf(A2, P2) & SamePaper(P1, P2) -> SameAuthor(A1, A2)
```

Blocking - How to Make Blocks

- How can we block authors?
- Need to tradeoff:
    - **Speed**
    - **Recall**
    - **Precision**

Blocking - How to Make Blocks

- How can we block authors?
- Need to tradeoff:
    - **Speed**
    - **Recall**
    - **Precision**
- Alphabetized Initials?

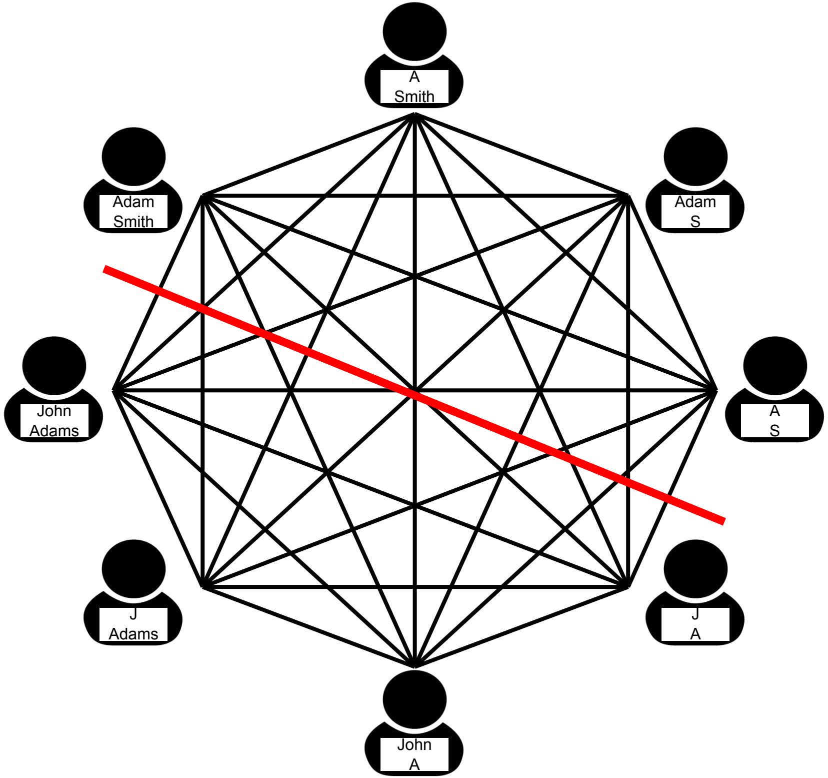
Pros:

- Fast
- Catch most misspellings
- Catch initials
- Catch Different Order
- Catch some nicknames

Cons:

- Miss some nicknames
- Miss totally different names

Blocking



Results - Quality

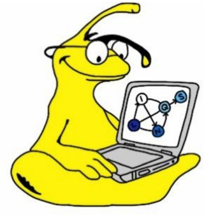
Size	**Transitive Relational**	**Blocking?**	**Time (sec)**	**Author F1**
Medium	None	No	166	0.7996
Medium	Equality	Yes	176	0.8157
Medium	Coauthor	Yes	173	0.8113
Medium	Paper	Yes	166	0.8158
Medium	All	Yes	180	**0.8467**

Results - Speed

Size	# Ground Rules	Transitive Relational	Blocking?	Time (sec)	Author F1
Small	220 M	Equality	No	21600+	N/A
Small	0.5 M	All	Yes	55	0.80946
Medium	1.5 M	All	Yes	180	0.846722
Large	3.3 M	All	Yes	413	0.734253

Additional Topics

Eriq Augustine and Golnoosh Farnadi
UC Santa Cruz
MLTrain 2018

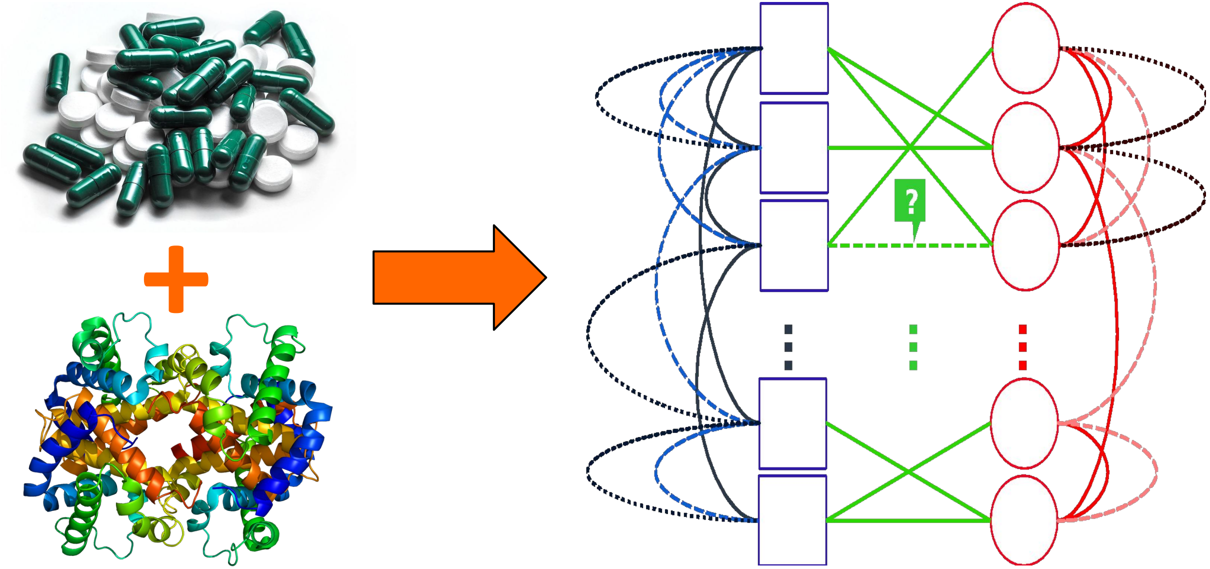


psl.linqs.org
github.com/linqs/psl

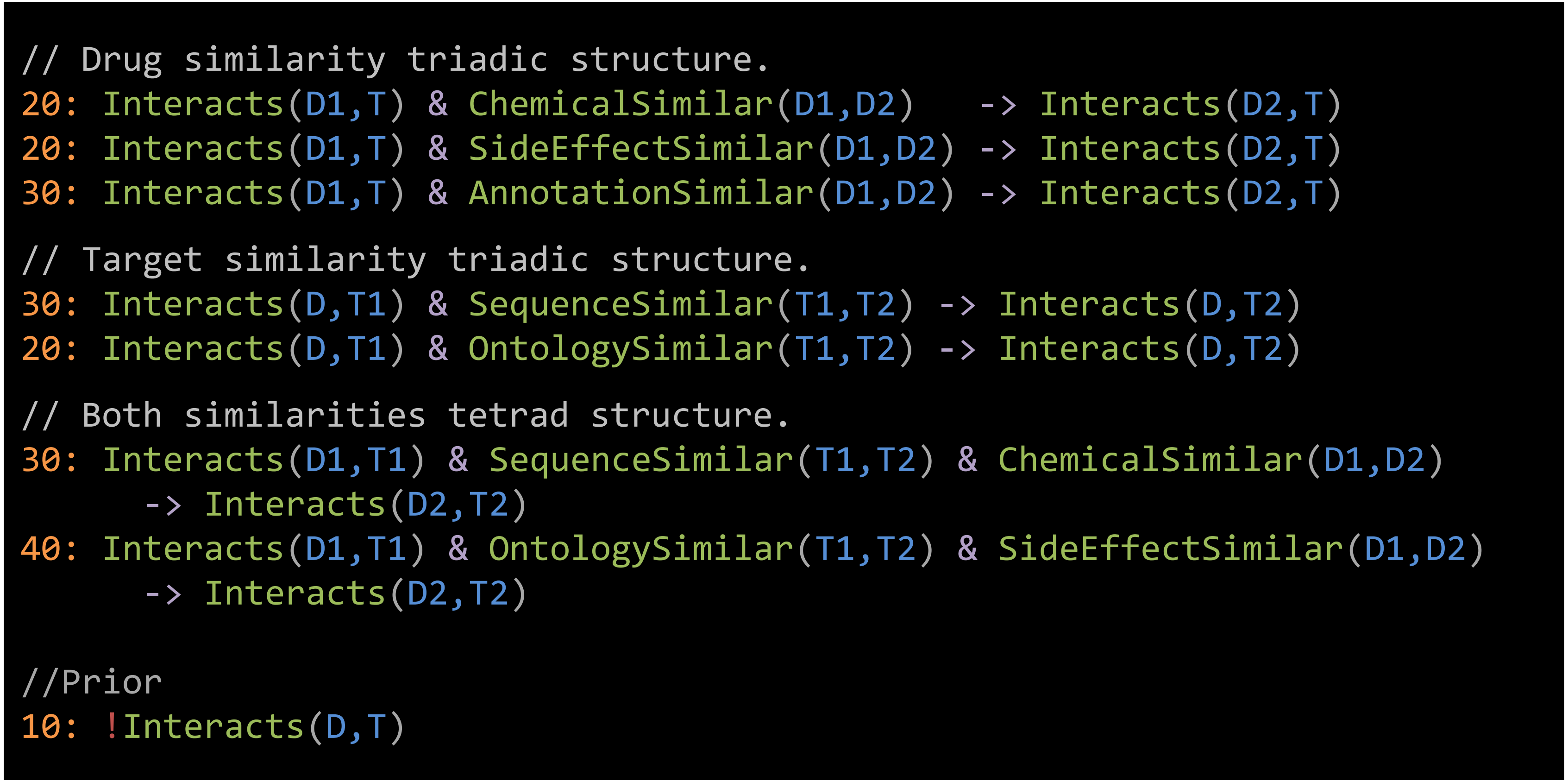
Additional PSL Models

Model - Drug Interaction Discovery

Predicting new drug-protein interactions for drug discovery, repurposing, side-effect prediction, and personalized medicine.



Model - Drug Interaction Discovery



```
// Drug similarity triadic structure.
20: Interacts(D1,T) & ChemicalSimilar(D1,D2)   -> Interacts(D2,T)
20: Interacts(D1,T) & SideEffectSimilar(D1,D2) -> Interacts(D2,T)
30: Interacts(D1,T) & AnnotationSimilar(D1,D2) -> Interacts(D2,T)

// Target similarity triadic structure.
30: Interacts(D,T1) & SequenceSimilar(T1,T2) -> Interacts(D,T2)
20: Interacts(D,T1) & OntologySimilar(T1,T2) -> Interacts(D,T2)

// Both similarities tetrad structure.
30: Interacts(D1,T1) & SequenceSimilar(T1,T2) & ChemicalSimilar(D1,D2)
      -> Interacts(D2,T2)
40: Interacts(D1,T1) & OntologySimilar(T1,T2) & SideEffectSimilar(D1,D2)
      -> Interacts(D2,T2)


//Prior
10: !Interacts(D,T)
```

https://github.com/shobeir/fakhraei_tcbb2014

Model - Drug Interaction Discovery

Task: Find new interactions between drugs and proteins targets in the drugbank dataset.

Newly Discovered Interactions

DRUGBANK
Open Data Drug & Drug Target Database

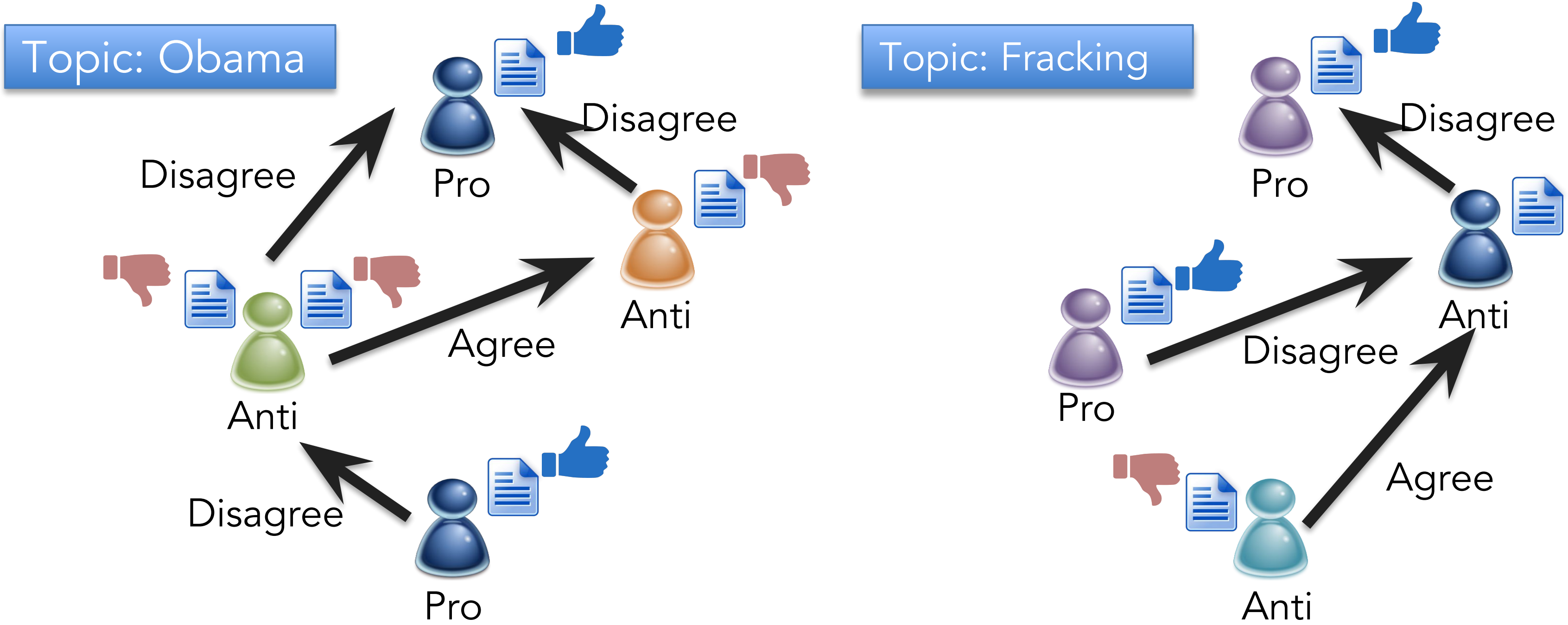
	AUC	AUPR	P@130
Perlman et al.	0.921	0.309	0.393
PSL-Model	0.926	0.344	**0.460**

Found 197 out of 78,750 possible interactions!

Network-based Drug-Target Interaction Prediction with Probabilistic Soft Logic, S. Fakhraei, B. Huang, L. Raschid, and L. Getoor, IEEE Transactions on Computational Biology and Bioinformatics (IEEE-TCBB), 2014. (Cover)
https://linqs.soe.ucsc.edu/node/9

Model - Debate Stance Classification

Jointly infer users' attitude on topics and polarity of interaction from online debate forum threads.



Joint Models of Disagreement and Stance, Sridhar, Foulds, Huang, Getoor & Walker, Annual Meeting of the Association for Computational Linguistics (ACL), 2015
https://linqs.soe.ucsc.edu/node/258

Model - Debate Stance Classification

```
// Priors from local text classifiers
1:  PriorPro(U,T)                 ->  Pro(U,T)
1:  PriorDisagree(U1,U2)          ->  Disagrees(U1,U2)

// Rules for stance
5:  Disagrees(U1,U2) & Pro(U1,T)  ->  !Pro(U2,T)
5: !Disagrees(U1,U2) & Pro(U1,T)  ->   Pro(U2,T)

// Rules for disagreement
5:  Pro(U1,T)        & Pro(U1,T)  ->  !Disagrees(U1,U2)
5: !Pro(U1,U2)       & Pro(U1,T)  ->   Disagrees(U1,U2)
```

Model - Debate Stance Classification

Task: Predict post and user stance on topics from two online debate forums:

- 4Forums.com: ~300 users,~6000 posts
- CreateDebate.org: ~300 users, ~1200 posts

4Forums.com

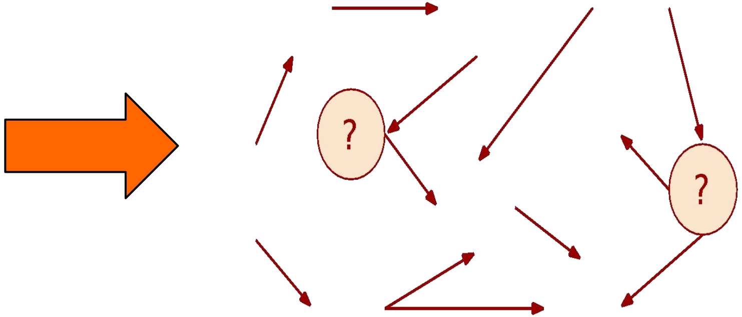
	User Stance Accuracy	Post Stance Accuracy
Logistic Regression Baseline	72.0	69.0
PSL-Post	73.7	72.5
PSL-Author	77.1	80.3

CreateDebate.org

	User Stance Accuracy	Post Stance Accuracy
Logistic Regression Baseline	70.2	62.7
PSL-Post	73.2	66.2
PSL-Author	74.0	72.7

Model - Finding Social Spammers

Find spammers in social media.



Collective Spammer Detection in Evolving Multi-Relational Social Networks, S. Fakhraei, J. Foulds, M. Shashanka, L. Getoor. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2015 https://linqs.soe.ucsc.edu/node/251

Model - Finding Social Spammers

```
// User generated reports
30: Credible(U1) & ReportedSpammer(U1,U2) -> Spammer(U2)

// Collective credibility
25: Spammer(U2)  & ReportedSpammer(U1,U2) -> Credible(U1)
25: !Spammer(U2) & ReportedSpammer(U1,U2) -> !Credible(U1)

// Prior credibility
20: PriorCredible(U)  -> Credible(U)
20: !PriorCredible(U) -> !Credible(U)

// Prior
10: !Spammer(U)
```

https://github.com/shobeir/fakhraei_kdd2015

Model – Spammer Detection

Task: Detecting social spammers in tagged.com social network using user-generated spammer reports.

- Attributes: Gender, Age, Account Age, Label
- Links: 8 Actions such as Like, Poke, Report Abuse, etc.

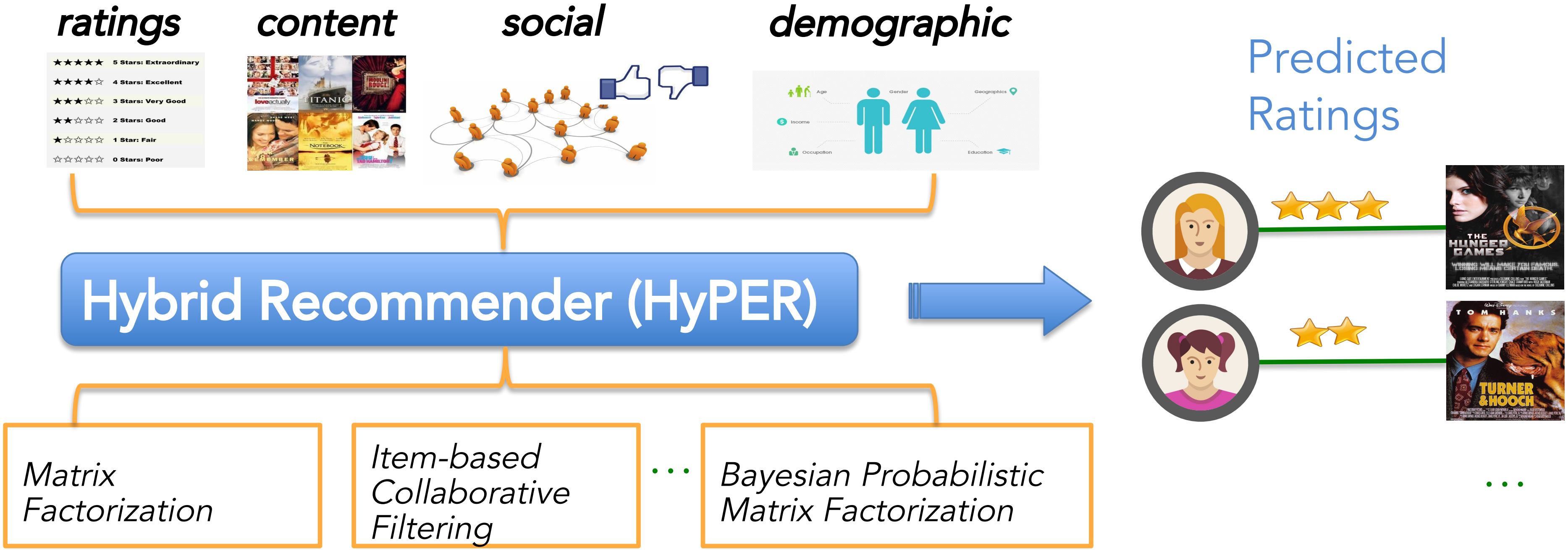
Spammers Detected

TAGGED

	AUC	AUPR
Using only reports	0.611	0.674
Using report and credibility	0.862	0.869
PSL (fully collective model)	0.873	0.884

Model - Hybrid Recommender Systems

Improve recommendations by combining data sources & recommenders.



ratings ***content*** ***social*** ***demographic***

★★★★★ 5 Stars: Extraordinary
★★★★☆ 4 Stars: Excellent
★★★☆☆ 3 Stars: Very Good
★★☆☆☆ 2 Stars: Good
★☆☆☆☆ 1 Star: Fair
☆☆☆☆☆ 0 Stars: Poor

Predicted Ratings

Hybrid Recommender (HyPER)

Matrix Factorization

Item-based Collaborative Filtering

…

Bayesian Probabilistic Matrix Factorization

…

HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems Kouki, Fakhraei, Foulds, Eirinaki, Getoor, RecSys15

https://linqs.soe.ucsc.edu/node/257

Model - Hybrid Recommender Systems

```
// Similar Items
10: Rating(U,I1) & PearsonSimilarityItems(I1,I2) -> Rating(U,I2)
10: Rating(U,I1) & ContentSimilarityItems(I1,I2) -> Rating(U,I2)

// Similar Users
10: Rating(U1,I) & PearsonSimilarityUsers(U1,U2) -> Rating(U2,I)
10: Rating(U1,I) & CosineSimilarityUsers (U1,U2) -> Rating(U2,I)

// Social Information
10: Friends(U1,U2) & Rating(U1,I) -> Rating(U2,I)

// Other Recommenders
10: MFRating(U,I)   -> Rating(U,I)
10: BPMFRating(U,I) -> Rating(U,I)

// Average Priors
1:  AvgUserRating(U) -> Rating(U,I)
1:  AvgItemRating(I) -> Rating(U,I)
```

https://github.com/pkouki/recsys2015

Model - Hybrid Recommender Systems

Task: Predict missing ratings

- Yelp: 34K users, 3.6K items, 99K ratings, 81K friendships, 500 business categories
- Last.fm: 1.8K users, 17K items, 92K ratings, 12K friendships, 9.7K artist tags

yelp

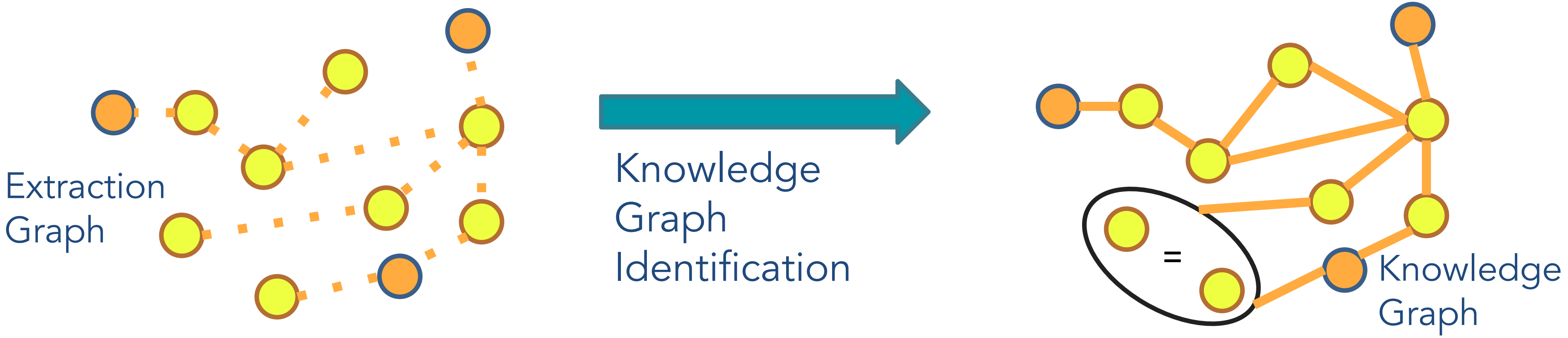
Model	RMSE
Item-based	1.216
MF	1.251
BPMF	1.191
Naïve Hybrid	1.179
BPMF-SRIC	1.191
HyPER	**1.173**

last·fm

Model	RMSE
Item-based	1.408
MF	1.178
BPMF	1.008
Naïve Hybrid	1.067
BPMF-SRIC	1.015
HyPER	**1.001**

Model - Knowledge Graph Identification

Refine noisy knowledge extractions into an accurate knowledge graph.



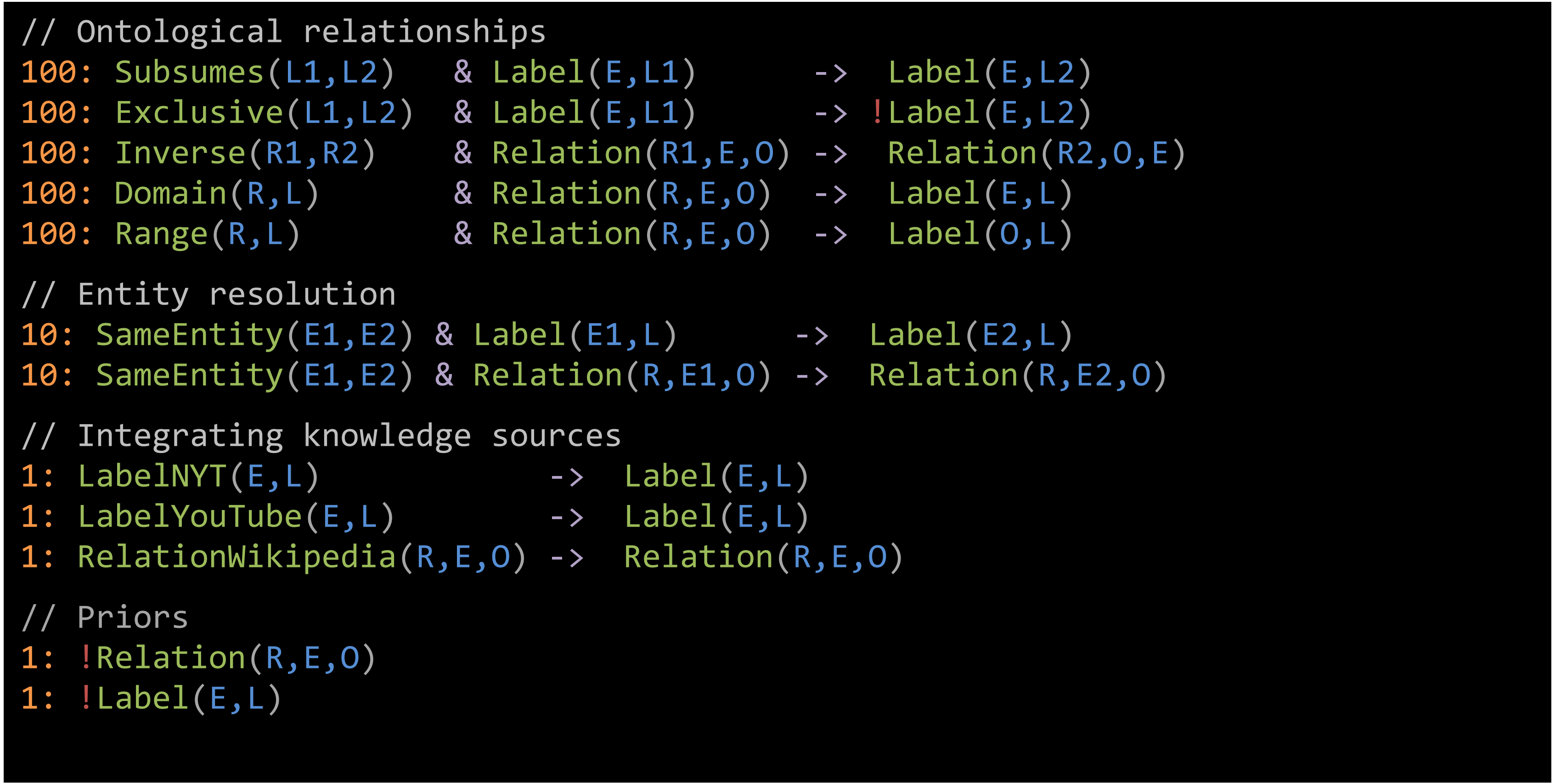
Extraction Graph

Knowledge Graph Identification

Knowledge Graph

Knowledge Graph Identification, Pujara, Miao, Getoor, & Cohen, ISWC, 2013
https://linqs.soe.ucsc.edu/node/28

Model - Knowledge Graph Identification



```
// Ontological relationships
100: Subsumes(L1,L2)   & Label(E,L1)        ->  Label(E,L2)
100: Exclusive(L1,L2)  & Label(E,L1)        -> !Label(E,L2)
100: Inverse(R1,R2)    & Relation(R1,E,O) ->  Relation(R2,O,E)
100: Domain(R,L)       & Relation(R,E,O)  ->  Label(E,L)
100: Range(R,L)        & Relation(R,E,O)  ->  Label(O,L)

// Entity resolution
10: SameEntity(E1,E2) & Label(E1,L)       ->  Label(E2,L)
10: SameEntity(E1,E2) & Relation(R,E1,O) ->  Relation(R,E2,O)

// Integrating knowledge sources
1: LabelNYT(E,L)            ->  Label(E,L)
1: LabelYouTube(E,L)        ->  Label(E,L)
1: RelationWikipedia(R,E,O) ->  Relation(R,E,O)

// Priors
1: !Relation(R,E,O)
1: !Label(E,L)
```

https://github.com/linqs/KnowledgeGraphIdentification

Model - Knowledge Graph Identification

Task: Construct a knowledge graph from millions of web text extractions from CMU's NELL project.

Knowledge graph for an explicit test set

	AUC	F1
Baseline	0.873	0.828
NELL	0.765	0.673
MLN (Jiang, 12)	0.899	0.836
PSL-KGI	0.904	0.853

Running Time: Inference completes in 10 seconds, produces **25K facts**

Complete knowledge graph including all NELL candidates

	AUC	F1
NELL	0.765	0.634
PSL-KGI	0.892	0.848

Running Time: Inference completes in 130 minutes, produces **4.3M facts**

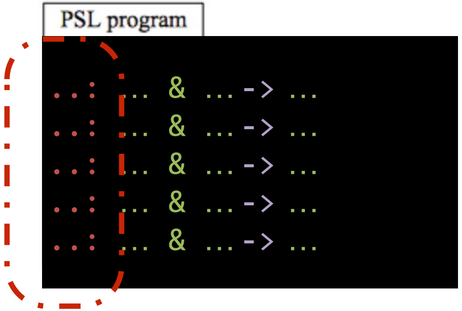
Using Statistics & Semantics to Turn Data Into Knowledge, Pujara, Miao, Getoor, & Cohen, AI Magazine, 2015
https://linqs.soe.ucsc.edu/node/272

Advanced Topics

Advanced Topics (not covered)

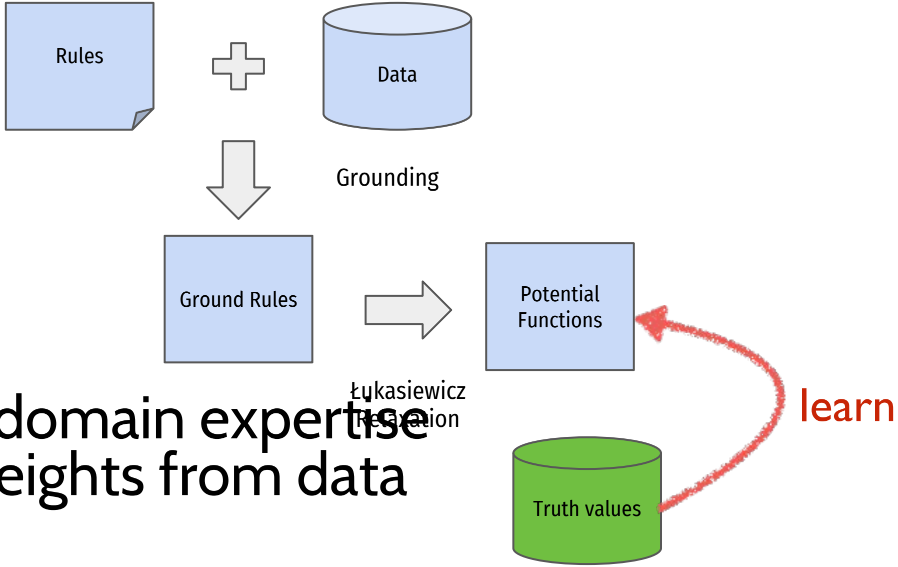
- Temporal & spatial modeling
- **Weight learning**
- **Structure learning**
- Causal modeling
- **Lifted Inference**
- **Fairness**
- Decision making

Weight Learning


PSL program

```
...: ... & ... -> ...
...: ... & ... -> ...
...: ... & ... -> ...
...: ... & ... -> ...
...: ... & ... -> ...
```

weight



- Manual weights given users’ domain expertise
- PSL supports learning rule weights from data

Weight Learning

- Maximum-likelihood Estimation: performs approximate maximum-likelihood estimation using MPE inference to approximate the gradient of the log-likelihood

$$\frac{\partial \log p(\mathbf{Y}|\mathbf{X})}{\partial \Lambda_q} = \mathbb{E}_\Lambda\left[\Phi_q(\mathbf{Y},\mathbf{X})\right] - \Phi_q(\mathbf{Y},\mathbf{X})$$

weight

expectation

Weight Learning

- **Maximum-pseudolikelihood Estimation:** which maximizes the likelihood of each variable conditioned on all other variables

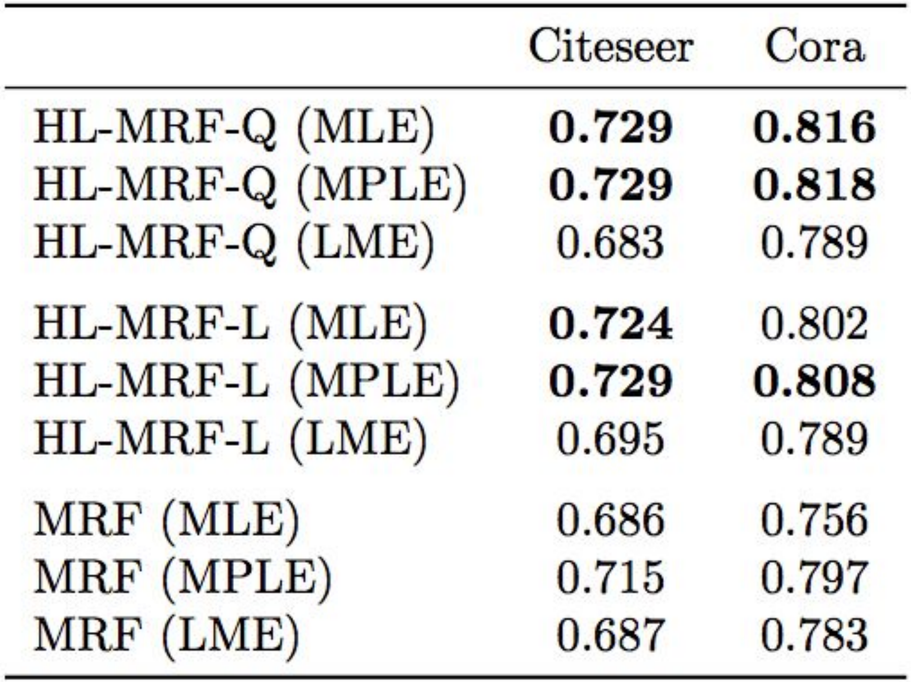
$$\frac{\partial \log P^*(Y|X)}{\partial \Lambda_q} = \sum_{i=1}^{n} \mathbb{E}_{Y_i|\mathrm{MB}}\left[\sum_{j\in t_q : i\in\phi_j} \phi_j(\mathbf{Y},\mathbf{X})\right] - \Phi_j(\mathbf{Y},\mathbf{X}).$$

- **Large Ma**

Markov blanket

$$\arg\min_{\tilde{\mathbf{Y}}} \Lambda^\top \Phi(\tilde{\mathbf{Y}},\mathbf{X}) - L(\mathbf{Y},\tilde{\mathbf{Y}}).$$

Result Highlights

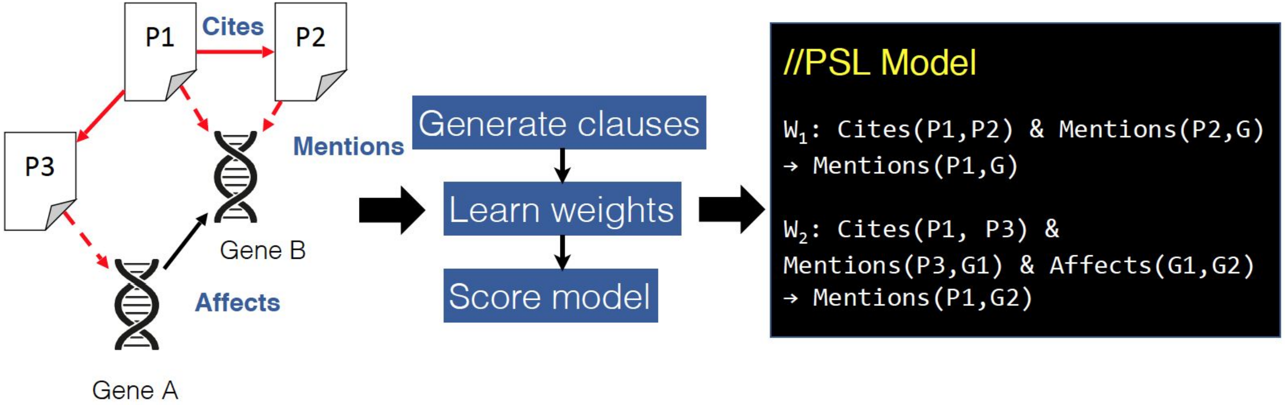


	Citeseer	Cora
HL-MRF-Q (MLE)	**0.729**	**0.816**
HL-MRF-Q (MPLE)	**0.729**	**0.818**
HL-MRF-Q (LME)	0.683	0.789
HL-MRF-L (MLE)	**0.724**	0.802
HL-MRF-L (MPLE)	**0.729**	**0.808**
HL-MRF-L (LME)	0.695	0.789
MRF (MLE)	0.686	0.756
MRF (MPLE)	0.715	0.797
MRF (LME)	0.687	0.783

S.H Bach, et. al, Hinge-loss Markov Random Fields: Convex Inference for Structured Prediction, UAI 2013

Structure Learning

- Learn weighted logical clauses from relational data



Generate clauses

Learn weights

Score model

//PSL Model

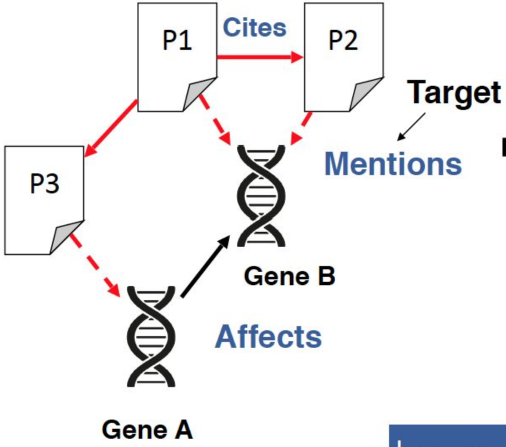
W_1: Cites(P1,P2) & Mentions(P2,G) → Mentions(P1,G)

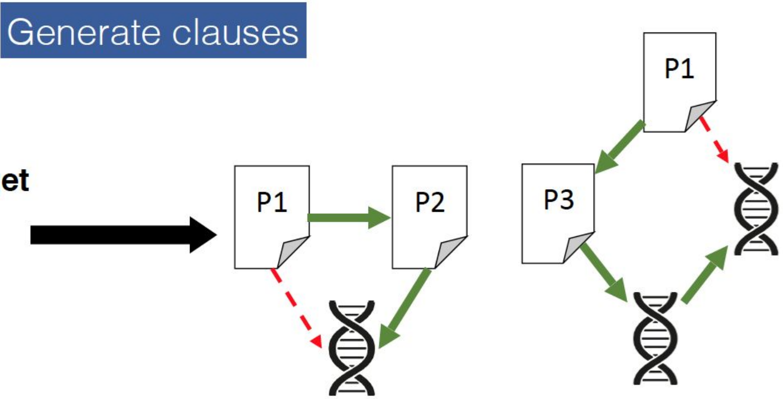
W_2: Cites(P1, P3) & Mentions(P3,G1) & Affects(G1,G2) → Mentions(P1,G2)

- **Challenges**: combinatorial clause search; repeated weight learning; intractable likelihood

Structure Learning in PSL

Generate clauses





BFS for path-constrained clauses

Learn weights Score model

W_1: Cites(P1,P2) & Mentions(P2,G) → Mentions(P1,G)
W_2: Cites(P1, P3) & Mentions(P3,G1) & Affects(G1,G2) → Mentions(P1,G2)

Piecewise pseudolikelihood scoring: only weight learning!

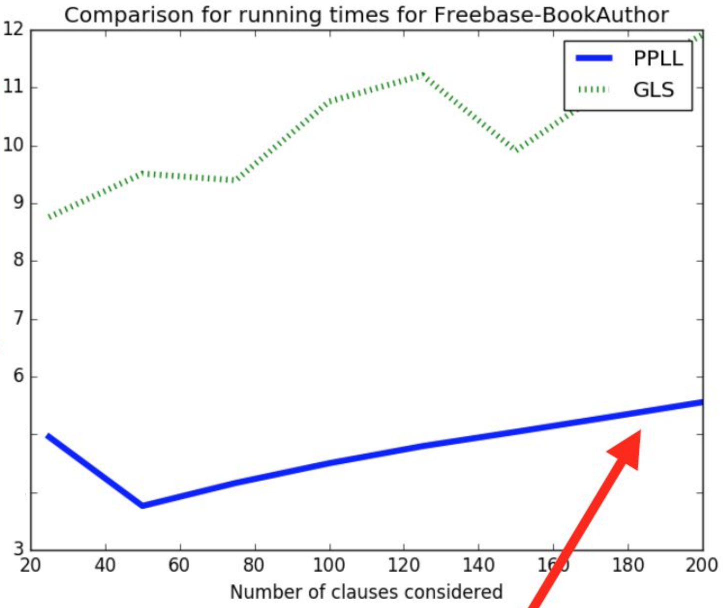
Result Highlights

5 fold CV:

Dataset	Greedy	PPLL
Fly	0.95	0.97
Yeast	0.86	0.90
DrugBank	0.66	0.76
Freebase	0.65	0.65

Significant AUC gains

Runtimes (in log sec):

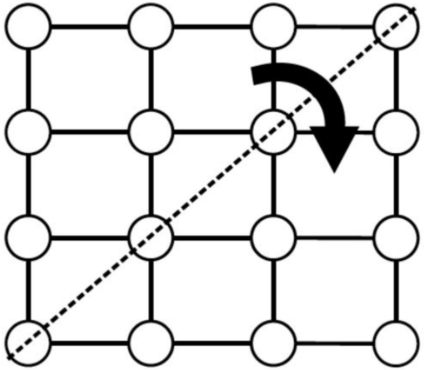


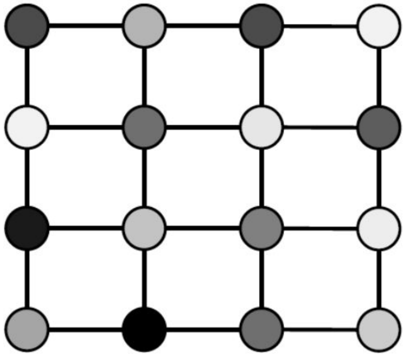
Scalability

Embar, Sridhar, Farnadi & Getoor, *StarAI* 2018

Lifted Inference in PSL

Lifted inference gives exponential speedups in symmetric graphical models.

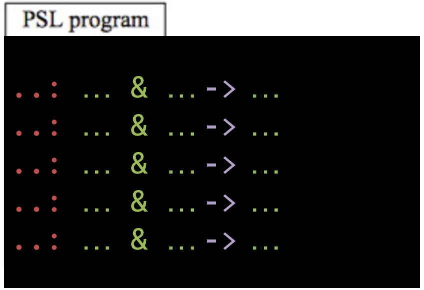




Existing lifted inference approaches focus on discrete graphical models

How to find symmetry in PSL with continuous atoms?

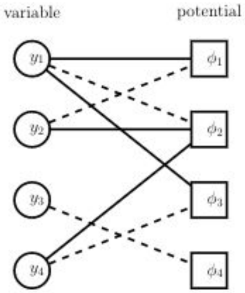
Lifted Inference in PSL



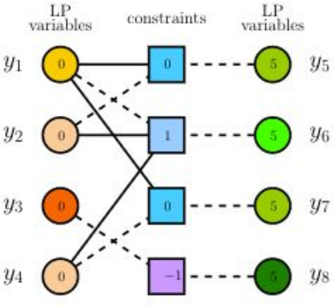
PSL program

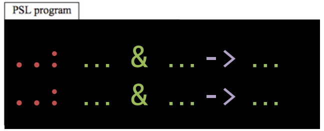
```
..: ... & ...-> ...
..: ... & ...-> ...
..: ... & ...-> ...
..: ... & ...-> ...
..: ... & ...-> ...
```

convert to a bipartite graph



color the graph with a color refinement algorithm



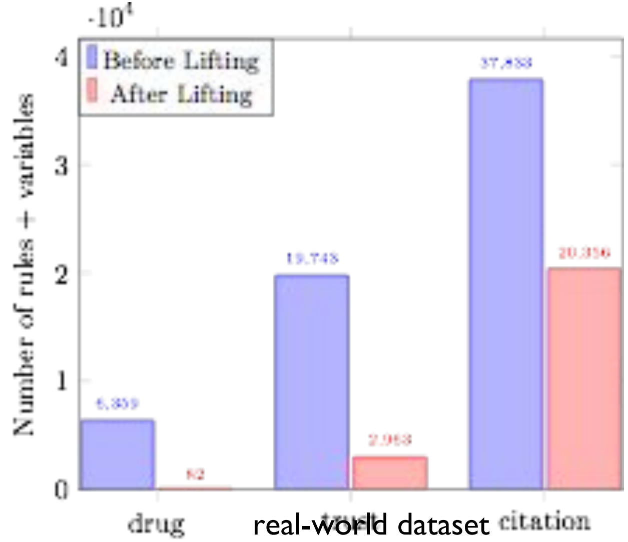


PSL program

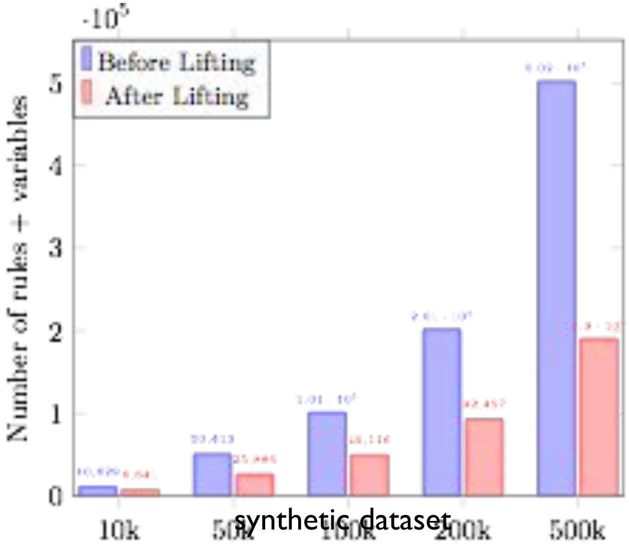
```
..: ... & ...-> ...
..: ... & ...-> ...
```

[in progress]

Result Highlights



real-world dataset



synthetic dataset

We observe a 3 to 68 times speed up in inference

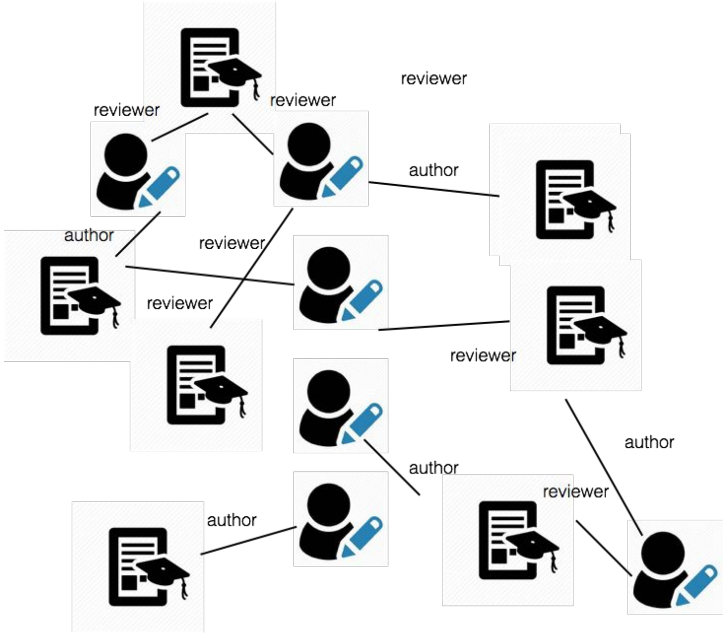
[in progress]

Fairness in Relational Domain

The goal of fairness-aware machine learning: is to ensure that the decisions made by an algorithm **do not discriminate against a population of individuals**

Challenge: Existing fairness approaches are based solely on attributes of individuals e.g., age, gender, race, etc.

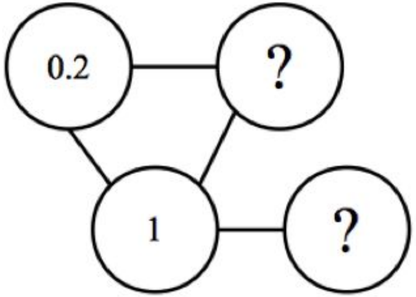
Our contribution: We introduce new notions of fairness that are able to capture the relational structure in a domain, e.g., citation network, corporate hierarchy, social network. etc.

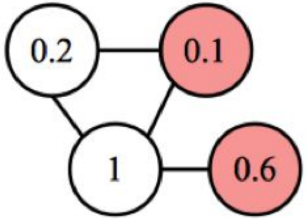


MAP Inference in PSL

$$p(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z(w,\mathbf{X})} \exp \sum_{j=1}^{m} w_j \phi_j$$

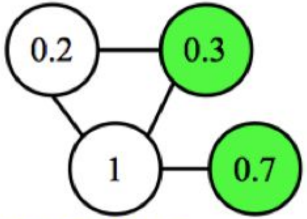
$$I_{MAP}(Y) = arg \max_{I(Y)} P(I(Y)|I(X))$$





highest probability
but (for some reason) unfair

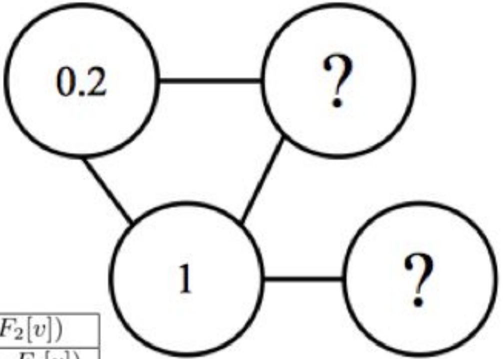
✗

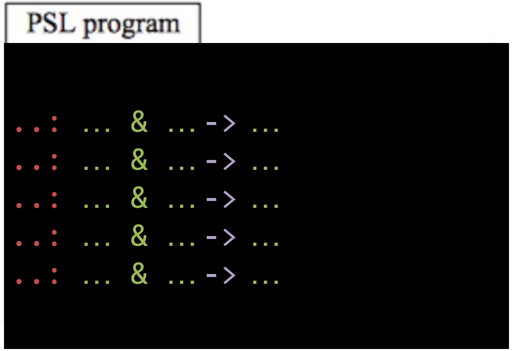


highest probability
among fair assignments

✓

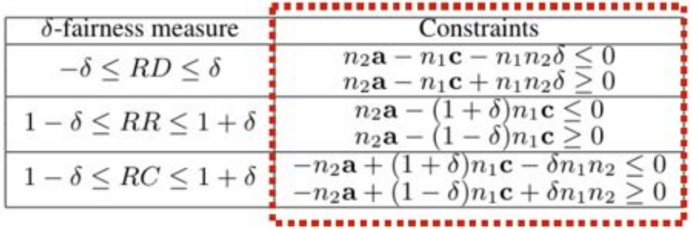
Fair MAP Inference in PSL



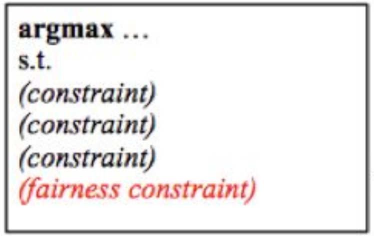

PSL program

```
..: ... & ... -> ...
..: ... & ... -> ...
..: ... & ... -> ...
..: ... & ... -> ...
..: ... & ... -> ...
```

a	$\sum_{v\in D_v} I(\neg d(v) \wedge F_1[v] \wedge F_2[v])$
c	$\sum_{v\in D_v} I(\neg d(v) \wedge F_1[v] \wedge \neg F_2[v])$
n_1	$\sum_{v\in D_v} I(F_1[v] \wedge F_2[v])$
n_2	$\sum_{v\in D_v} I(F_1[v] \wedge \neg F_2[v])$



δ-fairness measure	Constraints
$-\delta \leq RD \leq \delta$	$n_2\mathbf{a} - n_1\mathbf{c} - n_1 n_2 \delta \leq 0$ $n_2\mathbf{a} - n_1\mathbf{c} + n_1 n_2 \delta \geq 0$
$1-\delta \leq RR \leq 1+\delta$	$n_2\mathbf{a} - (1+\delta) n_1\mathbf{c} \leq 0$ $n_2\mathbf{a} - (1-\delta) n_1\mathbf{c} \geq 0$
$1-\delta \leq RC \leq 1+\delta$	$-n_2\mathbf{a} + (1+\delta) n_1\mathbf{c} - \delta n_1 n_2 \leq 0$ $-n_2\mathbf{a} + (1-\delta) n_1\mathbf{c} + \delta n_1 n_2 \geq 0$


argmax ...
s.t.
(constraint)
(constraint)
(constraint)
(fairness constraint)

Enforce fairness by adding extra linear constraints to the optimization problem

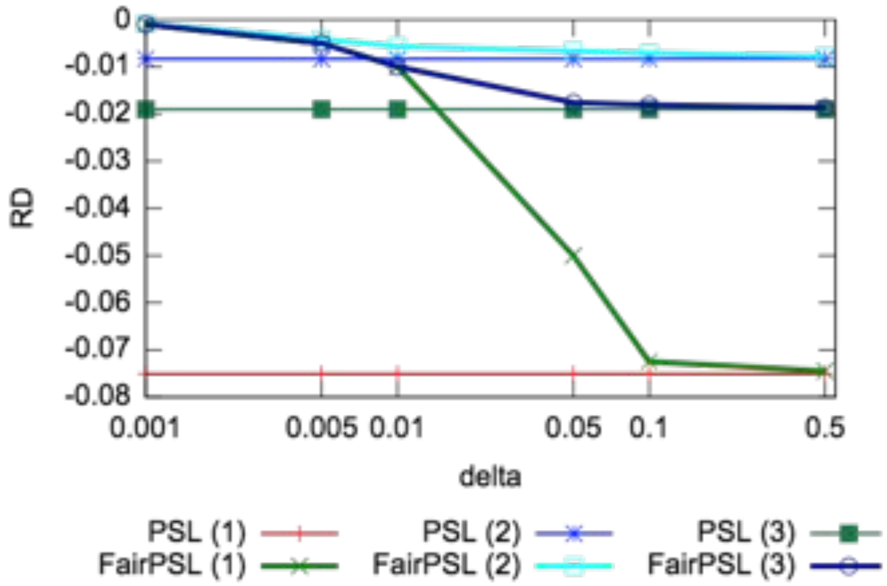
$$I_{MAP}(Y) = arg\max_{I(Y)} P(I(Y)|I(X))$$

Result Highlights

The paper reviewing problem: Ensure fair acceptance rate for students from high rank universities and low rank universities

We show our approach enforces fariness guarantees while preserving the accuracy of the predictions.

#1 has 102 papers, dataset #2 has 109 papers and dataset #3 has 101 papers
delta-fairness with five thresholds {0.001, 0.005, 0.01, 0.05, 0.1, 0.5}



The Code and data are available:
https://github.com/gfarnadi/FairPSL

Farnadi, Babaki & Getoor, *AAAI/ACM Conference on AI, Ethics, and Society* 2018

PSL Takeaways & Resources

PSL Takeaways

- Declarative language able to represent richly structured domains
- Supports collective reasoning – dependencies in inputs and outputs
- Mixes logical and probabilistic reasoning in flexible and scalable manner
- Applicable to wide variety of problems ranging from data integration & fusion to modeling socio-behavioral and scientific domains
- Eager to apply to additional domains, come talk with us if you are interested!

References

- Websites:
  - PSL: https://psl.linqs.org
  - LINQS: linqs.org
  - D3: https://d3.ucsc.edu
- Papers:
  - Main PSL Paper:
    Hinge-Loss Markov Random Fields and Probabilistic Soft Logic, Stephen Bach, Matthias Broecheler, Bert Huang, Lise Getoor, JMLR 2017
  - LINQS Publications: https://linqs.soe.ucsc.edu/biblio

Code

- Main Repository: https://github.com/linqs/psl
- Dev Repository: https://github.com/eriq-augustine/psl
- Examples: https://github.com/linqs/psl-examples
- Documentation:
    - API Reference: https://linqs-data.soe.ucsc.edu/psl-docs
    - Stable Wiki: https://github.com/linqs/psl/wiki
    - Development Wiki: https://github.com/eriq-augustine/psl/wiki

Thanks

- Dhanya Sridhar & Jay Pujara for slide material
- LINQS research group
- PSL Users & Contributors
- UCSC D3 Data Science Center Members
- Nick Vasiloglou II & Relational.AI
- UAI Organizers

Questions?