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 form 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.
Local sensors provide useful signals for prediction.
Relational Signals for Prediction

Sensors in nearby cities provide useful relational information.

San Jose

Santa Cruz

32 Miles
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

Santa Cruz

32 Miles
Diagram for Weather Forecasting

sensor

Santa Cruz

San Jose

sensor
Diagram for Weather Forecasting

Observed Value

Prediction
Diagram for Weather Forecasting

sensor → Santa Cruz

San Jose → sensor

SS → R_SC

R_SC → SS

RS_J → SS

SS → SJ
Local Predictive Model

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

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

$$\Pr(R_{SJ} | S_{SJ})$$
Incorrect Sensor Reading

Common problem: we get a faulty sensor reading.

Santa Cruz

-22°C

S_s

°C
Incorrect Local Predictions

$-22^\circ C$  

$S_S$  

$C$  

$R_{SC}$  

$R_{SJ}$  

$S_S$  

$J$
Incorrect Local Predictions

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

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

Common outcome: local model makes incorrect prediction.
Recall: sensors in nearby cities provide useful relational information!
Leveraging Relational Signals

-22°C $S_S$ $\xleftarrow{C} R_{SC}$

24°C $R_{SJ} \xrightarrow{S_S} J$
Leveraging Relational Signals

Distance variable captures closeness between cities.

-22°C $S_S$ $R_{SC}$ $D_{SC-S}$ $R_{SJ}$ $S_S$ 24°C

32 Miles
Leveraging Relational Signals

Distance variable captures closeness between cities.

\[
\Pr(R_{SC}, R_{SJ} | S_{SC}, S_{SJ}, D_{SC-SJ})
\]

-22°C

32 Miles

24°C
Leveraging Relational Signals

Joint modeling: forecasts in nearby cities should be similar.

\[
\Pr(R_{SC}, R_{SJ} | S_{SC}, S_{SJ}, D_{SC-SJ})
\]

-22°C \( S_S \) \( R_{SC} \) \( D_{SC-S} \) \( R_{SJ} \) \( S_S \) 24°C

32 Miles

Pr\((R_{SC}, R_{SJ})\)
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

San Diego

32 Miles

460 Miles
Relative Influences of Neighbors

-22°C

460 Miles

25°C

32 Miles
Relative Influences of Neighbors

Strength of collective influence depends on distance between cities.

-22°C

460 Miles

25°C

32 Miles

24°C
Relative Influences of Neighbors

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

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

Pr($R_{SC}, R_{SJ}, R_{SD} | S_{SC}, S_{SJ}, S_{SD}, D_{SC-SJ}, D_{SC-SD}$)
Markov Random Fields (MRFs)

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

\[ \text{Pr}(R_{SC}, R_{SJ}, R_{SD} | S_{SC}, S_{SJ}, S_{SD}, D_{SC-SJ}, D_{SC-SD}) \]
PSL - Syntax and Semantics
PSL uses first order logic-like rules.

5.0: \text{Rainy(City1)} \land \text{Distance(City1, City2)} \implies \text{Rainy(City2)}

1.0: \text{SenseRain(City)} \implies \text{Rainy(City)}
PSL uses first order logic-like rules.

5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
1.0: SenseRain(City) -> Rainy(City)
5.0: \text{Rainy}(\text{City1}) \& \text{Distance}(\text{City1}, \text{City2}) \rightarrow \text{Rainy}(\text{City2})

1.0: \text{SenseRain}(\text{City}) \rightarrow \text{Rainy}(\text{City})
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')
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)

![PSL Diagram]
\[ P(Y|X) \propto e^{\sum_{i=1}^{G} w_i \phi_i} \]

- Sum over all ground rules.
- The weight for a rule.
- The "satisfaction" of a ground rule. 1/0 for discrete logic.
\[ P(Y|X) \propto \exp\left( \sum_{i}^{G} w_i \phi_i \right) \]

\[ \text{argmax}_X \sum_{i}^{G} w_i \phi_i \]
Discrete MRF Inference == Weighted MAX-SAT == NP-Hard

$$\arg\max_x \sum_{i=1}^{G} w_i \phi_i$$
PSL - Continuous Relaxation

Relax "hard" satisfiability of each rule.

5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
First convert the rule to Disjunctive Normal Form.

5.0: $\text{Rainy}(\text{City1}) \land \text{Distance}(\text{City1}, \text{City2}) \rightarrow \text{Rainy}(\text{City2})$

$\text{Rainy}(\text{City1}) \land \text{Distance}(\text{City1}, \text{City2}) \rightarrow \text{Rainy}(\text{City2})$

$\neg(\text{Rainy}(\text{City1}) \land \text{Distance}(\text{City1}, \text{City2})) \lor \text{Rainy}(\text{City2})$

$\neg\text{Rainy}(\text{City1}) \lor \neg\text{Distance}(\text{City1}, \text{City2}) \lor \text{Rainy}(\text{City2})$
Use Łukasiewicz logic to relax hard logical operators.

- \( P \land Q = \max(0.0, P + Q - 1.0) \)
- \( P \lor Q = \min(1.0, P + Q) \)
- \( \neg Q = 1.0 - Q \)
Apply Łukasiewicz logic.

\[-\text{Rainy}(\text{City}1) \lor -\text{Distance}(\text{City}1, \text{City}2) \lor \text{Rainy}(\text{City}2)\]

\[\min(1.0, -\text{Rainy}(\text{City}1) + -\text{Distance}(\text{City}1, \text{City}2)) \lor \text{Rainy}(\text{City}2)\]

\[\min(1.0, -\text{Rainy}(\text{City}1) + -\text{Distance}(\text{City}1, \text{City}2) + \text{Rainy}(\text{City}2)\]

\[\min(1.0, (1.0 - \text{Rainy}(\text{City}1)) + (1.0 - \text{Distance}(\text{City}1, \text{City}2)) + \text{Rainy}(\text{City}2)\]

\[\min(1.0, 2.0 - (\text{Rainy}(\text{City}1) + \text{Distance}(\text{City}1, \text{City}2)) + \text{Rainy}(\text{City}2)\]
Apply Łukasiewicz logic to form a Hinge-Loss MRF.

Satisfaction:
\[
\min(1.0, 2.0 - (\text{Rainy}(\text{City1}) + \text{Distance}(\text{City1}, \text{City2})) + \text{Rainy}(\text{City2}))
\]

Distance to satisfaction:
\[
1.0 - \min(1.0, 2.0 - (\text{Rainy}(\text{City1}) + \text{Distance}(\text{City1}, \text{City2})) + \text{Rainy}(\text{City2}))
\]
HL-MRF Inference == Sum of Convex Function == Convex!
Solve with Alternating Direction Method of Multipliers (ADMM)

\[
\underset{x}{\text{argmax}} \sum_{i}^{G} w_i \phi_i
\]

https://web.stanford.edu/~boyd/admm.html
PSL - Rules to Assignments

1. Rules
2. Data
3. Grounding
   - Ground Rules
   - Łukasiewicz Relaxation
4. Potential Functions
5. Inference
6. Random Variable 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](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?

\[ Is(A, U, L) \]

Attribute “A” of User “U” is Labeled “L”
What is Collective Classification?

example:

Is(Gender,Alice,Female)

Is(Age,Bob,Young)

Is(Personality,Carol,Introvert)
Local Predictor Rule

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

Predicts$(S,A,U,L)$ -> Is$(A,U,L)$
Local Predictor Rule

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

\[
P(L|T)
\]

<table>
<thead>
<tr>
<th>(T_U)</th>
<th>(L_U)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Local Predictor Rule

\[ \text{Predicts}(\text{Txt, Personality, Alice, Int}) \rightarrow \text{Is} (\text{Personality, Alice, Int}) \]

Example:

Extrovert

Introvert

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

\[
\begin{array}{ccc|c}
\hline
u_1 & u_2 & \ldots & u_n \\ 
\hline
u_2 & u_3 & \ldots & \ldots \\ 
\hline
u_n & u_{2n} & \ldots & \ldots \\ 
\hline
\end{array}
\]
Collective Rule (1/3) (user-user relations)

- Friend
- Follower
- Neighbour
- Spouse
- Idol
- Coauthor
- Colleague
Collective Rule (1/3) (user-user relations)

\[ \text{Friend}(U_1, U_2) \land \text{Is}(A, U_1, L) \rightarrow \text{Is}(A, U_2, L) \]
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:

\[ \text{Friend}(Alice, Carol) \land \text{Is(Personality, Carol, Ext)} \rightarrow \text{Is(Personality, Alice, Ext)} \]
Collective Rule (2/3) (user-item relations)

- Page likes
- Item rating
- Movie ratings

matrix-factorisation
**Collective Rule (2/3) (user-item relations)**

- Page likes
- Item rating
- Movie ratings

| u1  | I1 |   |
| u2  | I7 |   |
| ... | ... |   |
| un  | I8 |   |

**matrix-factorisation**
Collective Rule (2/3) (user-item relations)

\[ \text{Likes}(U_1, I) \land \text{Likes}(U_2, I) \land \text{Is}(A, U_1, L) \rightarrow \text{Is}(A, U_2, L) \]
Collective Rule (2/3) (user-item relations)

example:

\[ \text{Likes}(\text{Alice, Party}) \land \text{Likes}(\text{Carol, Party}) \land \text{Is(Personality, Carol, Ext}) \rightarrow \text{Is(Personality, Alice, Ext)} \]
Collective Rule (3/3) (user-group relations)

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

\[ \text{Joins}(U_1, G) \land \text{Joins}(U_2, G) \land \text{Is}(A, U_1, L) \rightarrow \text{Is}(A, U_2, L) \]
Collective Rule (3/3) (user-group relations)

example:

\[\text{Joins}(\text{Carol}, \text{Action-Movies}) \land \text{Joins}(\text{Alice}, \text{Action-Movie}) \land \text{Is}(\text{Personality}, \text{Carol}, \text{Ext}) \rightarrow \text{Is}(\text{Personality}, \text{Alice}, \text{Ext})\]
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,\text{\texttt{+L}}) = 1
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
// 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
// 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
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
## Evaluation Result (Personality prediction-synthetic data)

<table>
<thead>
<tr>
<th>Type of the model</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.5</td>
</tr>
<tr>
<td>Local Predictor</td>
<td>0.811655</td>
</tr>
<tr>
<td>Friendship links</td>
<td>0.528963</td>
</tr>
<tr>
<td>Likes</td>
<td>0.724014</td>
</tr>
<tr>
<td>Joins</td>
<td>0.865880</td>
</tr>
<tr>
<td>All</td>
<td>0.777315</td>
</tr>
</tbody>
</table>

**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)

<table>
<thead>
<tr>
<th>Type of the model</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.5</td>
</tr>
<tr>
<td>Local Predictor</td>
<td>0.811655</td>
</tr>
<tr>
<td>Friendship links</td>
<td>0.528963</td>
</tr>
<tr>
<td>Likes</td>
<td>0.724014</td>
</tr>
<tr>
<td>Joins</td>
<td>0.865880</td>
</tr>
<tr>
<td>All</td>
<td>0.777315</td>
</tr>
<tr>
<td>All</td>
<td>0.913516</td>
</tr>
</tbody>
</table>

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]
### 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

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

<table>
<thead>
<tr>
<th>Model/Characteristic</th>
<th>Gender</th>
<th>Age</th>
<th>Opn</th>
<th>Con</th>
<th>Ext</th>
<th>Agr</th>
<th>Neu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.492</td>
<td>0.488</td>
<td>0.502</td>
<td>0.502</td>
<td>0.506</td>
<td>0.504</td>
<td>0.486</td>
</tr>
<tr>
<td>PSL-Textual</td>
<td>0.650</td>
<td>0.710</td>
<td>0.570</td>
<td>0.567</td>
<td>0.553</td>
<td>0.550</td>
<td>0.542</td>
</tr>
<tr>
<td>PSL-Visual</td>
<td>0.850</td>
<td>0.579</td>
<td>0.505</td>
<td>0.521</td>
<td>0.531</td>
<td>0.531</td>
<td>0.515</td>
</tr>
<tr>
<td>PSL-Relational</td>
<td>0.853</td>
<td>0.881</td>
<td>0.648</td>
<td>0.618</td>
<td>0.592</td>
<td>0.571</td>
<td>0.572</td>
</tr>
<tr>
<td>PSL-Fusion</td>
<td>0.893</td>
<td>0.893</td>
<td>0.654</td>
<td>0.622</td>
<td>0.599</td>
<td>0.581</td>
<td>0.58</td>
</tr>
</tbody>
</table>
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.”
PSL Structural Balance

//Rules for cycle structure
1: Trusts(A,B) & Trusts(B,C) -> Trusts(C,A)
1: !Trusts(A,B) & !Trusts(B,C) -> Trusts(C,A)

//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)
Model #2: Social status (Cosmides & Tooby, ’92). Strong ties indicate unidirectional respect, “looking up to,” expertise status.

Leskovec et al. (2010) explored occurrence of both models in data and single-edge prediction.
// Rules for cycle structure
1: \text{Trusts}(A,B) \land \text{Trusts}(B,C) \rightarrow \neg\text{Trusts}(C,A)
1: \neg\text{Trusts}(A,B) \land \neg\text{Trusts}(B,C) \rightarrow \text{Trusts}(C,A)

// Rules for Non-cycle structure
1: \text{Trusts}(A,B) \land \neg\text{Trusts}(C,B) \rightarrow \neg\text{Trusts}(C,A)
1: \neg\text{Trusts}(A,B) \land \text{Trusts}(C,B) \rightarrow \text{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(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(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(C, A) & 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(C, A) & !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

- Rules
- Data
- Grounding
- Ground Rules
- Łukasiewicz Relaxation
- Potential Functions
- Inference
- Random Variable Assignments
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
<table>
<thead>
<tr>
<th>Type of the model</th>
<th>AUC</th>
<th>AUC (positive)</th>
<th>AUC (negative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance Theory</td>
<td>0.808961</td>
<td>0.973752</td>
<td>0.450463</td>
</tr>
<tr>
<td>Social Status</td>
<td>0.633428</td>
<td>0.946462</td>
<td>0.231061</td>
</tr>
<tr>
<td>Latent Model</td>
<td>0.917668</td>
<td>0.991246</td>
<td>0.557115</td>
</tr>
</tbody>
</table>
## Evaluation Result (Trust prediction- Epinions data)

<table>
<thead>
<tr>
<th>Type of the model</th>
<th>AUC</th>
<th>AUC (positive)</th>
<th>AUC (negative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance Theory</td>
<td>0.808961</td>
<td>0.973752</td>
<td>0.450463</td>
</tr>
<tr>
<td>Social Status</td>
<td>0.633428</td>
<td>0.946462</td>
<td>0.231061</td>
</tr>
<tr>
<td>Latent Model</td>
<td>0.917668</td>
<td>0.991246</td>
<td>0.557115</td>
</tr>
</tbody>
</table>
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**
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

```bash
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

<table>
<thead>
<tr>
<th>Size</th>
<th>Authors</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1136</td>
<td>864</td>
</tr>
<tr>
<td>Medium</td>
<td>1813</td>
<td>1143</td>
</tr>
<tr>
<td>Large</td>
<td>2892</td>
<td>1504</td>
</tr>
</tbody>
</table>

Federalist No. 1

Federalist No. 10

James M.

Publius

Publius

Madison, J.
Initial Model

James M. → Federalist No. 1 → Madison, J.

Publius → Federalist No. 10 → Publius
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.

\[
\text{SameAuthor}(A_1, A_2) \\
\& \text{ SameAuthor}(A_2, A_3) \\
\Rightarrow \text{ SameAuthor}(A_1, A_3)
\]
What other transitive relational rules can we get?
Transitive Relational

Publius

Federalist No. 1

Alexander Hamilton

The First Federalist Paper
Transitive Relational

AuthorOf(A1, P1) & AuthorOf(A2, P2) & SamePaper(P1, P2) -> SameAuthor(A1, A2)
What other transitive relational rules can we get?
Transitive Relational

AuthorOf(A1, P1) & AuthorOf(A2, P2) & AuthorOf(CA1, P1) & AuthorOf(CA2, P2) & SameAuthor(CA1, CA2) -> SameAuthor(A1, A2)
Transitive Blowup!

$\text{SameAuthor}(A_1, A_2) \land \text{SameAuthor}(A_2, A_3) \rightarrow \text{SameAuthor}(A_1, A_3)$
Transitive Blowup!

Arbitrarily choose **three** authors.
Recall we have 1813 authors.

\[
\text{SameAuthor}(A_1, A_2) \land \text{SameAuthor}(A_2, A_3) \implies \text{SameAuthor}(A_1, A_3)
\]
Transitive Blowup!

Arbitrarily choose three authors. Recall we have 1813 authors.

\[
\binom{1813}{3} \\
\approx 1 \text{ Billion Ground Rules}
\]
● 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
AuthorBlock(A₁, B) & AuthorBlock(A₂, B) & AuthorBlock(A₃, B)
& SameAuthor(A₁, A₂) & SameAuthor(A₂, A₃) -> SameAuthor(A₁, A₃)
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
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
<table>
<thead>
<tr>
<th>Size</th>
<th>Transitive Relational</th>
<th>Blocking?</th>
<th>Time (sec)</th>
<th>Author F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>None</td>
<td>No</td>
<td>166</td>
<td>0.7996</td>
</tr>
<tr>
<td>Medium</td>
<td>Equality</td>
<td>Yes</td>
<td>176</td>
<td>0.8157</td>
</tr>
<tr>
<td>Medium</td>
<td>Coauthor</td>
<td>Yes</td>
<td>173</td>
<td>0.8113</td>
</tr>
<tr>
<td>Medium</td>
<td>Paper</td>
<td>Yes</td>
<td>166</td>
<td>0.8158</td>
</tr>
<tr>
<td>Medium</td>
<td>All</td>
<td>Yes</td>
<td>180</td>
<td>0.8467</td>
</tr>
</tbody>
</table>
## Results - Speed

<table>
<thead>
<tr>
<th>Size</th>
<th># Ground Rules</th>
<th>Transitive Relational</th>
<th>Blocking?</th>
<th>Time (sec)</th>
<th>Author F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>220 M</td>
<td>Equality</td>
<td>No</td>
<td>21600+</td>
<td>N/A</td>
</tr>
<tr>
<td>Small</td>
<td>0.5 M</td>
<td>All</td>
<td>Yes</td>
<td>55</td>
<td>0.80946</td>
</tr>
<tr>
<td>Medium</td>
<td>1.5 M</td>
<td>All</td>
<td>Yes</td>
<td>180</td>
<td>0.846722</td>
</tr>
<tr>
<td>Large</td>
<td>3.3 M</td>
<td>All</td>
<td>Yes</td>
<td>413</td>
<td>0.734253</td>
</tr>
</tbody>
</table>
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: \text{Interacts}(D_1,T) \& \text{ChemicalSimilar}(D_1,D_2) \rightarrow \text{Interacts}(D_2,T)
20: \text{Interacts}(D_1,T) \& \text{SideEffectSimilar}(D_1,D_2) \rightarrow \text{Interacts}(D_2,T)
30: \text{Interacts}(D_1,T) \& \text{AnnotationSimilar}(D_1,D_2) \rightarrow \text{Interacts}(D_2,T)

// Target similarity triadic structure.
30: \text{Interacts}(D,T_1) \& \text{SequenceSimilar}(T_1,T_2) \rightarrow \text{Interacts}(D,T_2)
20: \text{Interacts}(D,T_1) \& \text{OntologySimilar}(T_1,T_2) \rightarrow \text{Interacts}(D,T_2)

// Both similarities tetrad structure.
30: \text{Interacts}(D_1,T_1) \& \text{SequenceSimilar}(T_1,T_2) \& \text{ChemicalSimilar}(D_1,D_2) \\
    \rightarrow \text{Interacts}(D_2,T_2)
40: \text{Interacts}(D_1,T_1) \& \text{OntologySimilar}(T_1,T_2) \& \text{SideEffectSimilar}(D_1,D_2) \\
    \rightarrow \text{Interacts}(D_2,T_2)

//Prior
10: !\text{Interacts}(D,T)

https://github.com/shobeir/fakhraei_tcbb2014
Task: Find new interactions between drugs and proteins targets in the drugbank dataset.

Newly Discovered Interactions

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC</th>
<th>AUPR</th>
<th>P@130</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perlman et al.</td>
<td>0.921</td>
<td>0.309</td>
<td>0.393</td>
</tr>
<tr>
<td>PSL-Model</td>
<td>0.926</td>
<td>0.344</td>
<td>0.460</td>
</tr>
</tbody>
</table>

Found 197 out of 78,750 possible interactions!


https://lings.soee.ucsc.edu/node/9
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](https://linqs.soe.ucsc.edu/node/258)
// 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

<table>
<thead>
<tr>
<th></th>
<th>4Forums.com</th>
<th>CreateDebate.org</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User Stance Accuracy</td>
<td>Post Stance Accuracy</td>
</tr>
<tr>
<td>Logistic Regression Baseline</td>
<td>72.0</td>
<td>69.0</td>
</tr>
<tr>
<td>PSL-Post</td>
<td>73.7</td>
<td>72.5</td>
</tr>
<tr>
<td>PSL-Author</td>
<td>77.1</td>
<td>80.3</td>
</tr>
</tbody>
</table>
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://lings.soe.ucsc.edu/node/251](https://lings.soe.ucsc.edu/node/251)
// 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.

<table>
<thead>
<tr>
<th>Spammers Detected</th>
<th>AUC</th>
<th>AUPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using only reports</td>
<td>0.611</td>
<td>0.674</td>
</tr>
<tr>
<td>Using report and credibility</td>
<td>0.862</td>
<td>0.869</td>
</tr>
<tr>
<td>PSL (fully collective model)</td>
<td>0.873</td>
<td>0.884</td>
</tr>
</tbody>
</table>
Model - Hybrid Recommender Systems

Improve recommendations by combining data sources & recommenders.

HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems
Kouki, Fakhraei, Foulds, Eirinaki, Getoor, RecSys15
https://lings.soe.ucsc.edu/node/257
// 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

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item-based</td>
<td>1.216</td>
</tr>
<tr>
<td>MF</td>
<td>1.251</td>
</tr>
<tr>
<td>BPMF</td>
<td>1.191</td>
</tr>
<tr>
<td>Naïve Hybrid</td>
<td>1.179</td>
</tr>
<tr>
<td>BPMF-SRIC</td>
<td>1.191</td>
</tr>
<tr>
<td><strong>HyPER</strong></td>
<td><strong>1.173</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item-based</td>
<td>1.408</td>
</tr>
<tr>
<td>MF</td>
<td>1.178</td>
</tr>
<tr>
<td>BPMF</td>
<td>1.008</td>
</tr>
<tr>
<td>Naïve Hybrid</td>
<td>1.067</td>
</tr>
<tr>
<td>BPMF-SRIC</td>
<td>1.015</td>
</tr>
<tr>
<td><strong>HyPER</strong></td>
<td><strong>1.001</strong></td>
</tr>
</tbody>
</table>
Refine noisy knowledge extractions into an accurate knowledge graph.

Knowledge Graph Identification, Pujara, Miao, Getoor, & Cohen, ISWC, 2013
https://linqs.soe.ucsc.edu/node/28
// 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
Task: Construct a knowledge graph from millions of web text extractions from CMU’s NELL project.

**Knowledge graph for an explicit test set**

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.873</td>
<td>0.828</td>
</tr>
<tr>
<td>NELL</td>
<td>0.765</td>
<td>0.673</td>
</tr>
<tr>
<td>MLN (Jiang, 12)</td>
<td>0.899</td>
<td>0.836</td>
</tr>
<tr>
<td>PSL-KGI</td>
<td>0.904</td>
<td>0.853</td>
</tr>
</tbody>
</table>

**Complete knowledge graph including all NELL candidates**

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NELL</td>
<td>0.765</td>
<td>0.634</td>
</tr>
<tr>
<td>PSL-KGI</td>
<td>0.892</td>
<td>0.848</td>
</tr>
</tbody>
</table>

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

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


[https://linqs.soe.ucsc.edu/node/272](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

- Manual weights given users’ domain expertise
- PSL supports learning rule weights from data
• **Maximum-likelihood Estimation:** performs approximate maximum-likelihood estimation using MPE inference to approximate the gradient of the log-likelihood

\[
\frac{\partial \log p(Y|X)}{\partial \Lambda_q} = E_\Lambda [\Phi_q(Y, X)] - \Phi_q(Y, X)
\]

- weight
- expectation
Weight Learning

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

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

- **Large Markov blanket**
## Result Highlights

<table>
<thead>
<tr>
<th>Model</th>
<th>CiteSeer</th>
<th>Cora</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL-MRF-Q (MLE)</td>
<td>0.729</td>
<td>0.816</td>
</tr>
<tr>
<td>HL-MRF-Q (MPLE)</td>
<td>0.729</td>
<td>0.818</td>
</tr>
<tr>
<td>HL-MRF-Q (LME)</td>
<td>0.683</td>
<td>0.789</td>
</tr>
<tr>
<td>HL-MRF-L (MLE)</td>
<td>0.724</td>
<td>0.802</td>
</tr>
<tr>
<td>HL-MRF-L (MPLE)</td>
<td>0.729</td>
<td>0.808</td>
</tr>
<tr>
<td>HL-MRF-L (LME)</td>
<td>0.695</td>
<td>0.789</td>
</tr>
<tr>
<td>MRF (MLE)</td>
<td>0.686</td>
<td>0.756</td>
</tr>
<tr>
<td>MRF (MPLE)</td>
<td>0.715</td>
<td>0.797</td>
</tr>
<tr>
<td>MRF (LME)</td>
<td>0.687</td>
<td>0.783</td>
</tr>
</tbody>
</table>

Structure Learning

- Learn weighted logical clauses from relational data

- **Challenges**: combinatorial clause search; repeated weight learning; intractable likelihood
Structure Learning in PSL

Generate clauses

BFS for path-constrained clauses

Learn weights

Score model

$W_1$: \text{Cites}(P1,P2) \& \text{Mentions}(P2,G) \rightarrow \text{Mentions}(P1,G)

$W_2$: \text{Cites}(P1, P3) \& \text{Mentions}(P3,G1) \& \text{Affects}(G1,G2) \rightarrow \text{Mentions}(P1,G2)

Piecewise pseudolikelihood scoring: only weight learning!
Result Highlights

5 fold CV:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Greedy</th>
<th>PPLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fly</td>
<td>0.95</td>
<td>0.97</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.86</td>
<td>0.90</td>
</tr>
<tr>
<td>DrugBank</td>
<td>0.66</td>
<td>0.76</td>
</tr>
<tr>
<td>Freebase</td>
<td>0.65</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Runtimes (in log sec):

Significant AUC gains

Embar, Sridhar, Farnadi & Getoor, StarAI 2018

Scalability
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

Convert to a bipartite graph

Color the graph with a color refinement algorithm

[in progress]
Result Highlights

We observe a 3 to 68 times speed up in inference
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.
\[ p(Y|X) = \frac{1}{Z(w, X)} \exp \sum_{j=1}^{m} w_j \phi_j \]

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

- Highest probability
- But (for some reason) unfair
- Highest probability among fair assignments
Fair MAP Inference in PSL

\[ I_{MAP}(Y) = \arg\max_{I(Y)} P(I(Y)|I(X)) \]
The paper reviewing problem: Ensure fair acceptance rate for students from high rank universities and low rank universities

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

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](https://psl.linqs.org)
  - LINQS: [linqs.org](http://linqs.org)
  - D3: [https://d3.ucsc.edu](https://d3.ucsc.edu)

- **Papers:**
  - **Main PSL Paper:**
  - LINQS Publications: [https://linqs.soe.ucsc.edu/biblio](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?