Programming with Personalized PageRank
A Locally Groundable First-Order Probabilistic Logic

William Yang Wang
Katie Mazaitis & William Cohen

School of Computer Science
Carnegie Mellon University
The Problem

• Task: learning to reason on large graphs.

• Approach: 1\textsuperscript{st}-order probabilistic logic inference.
Motivation

• The Issue:
  grounding with many inference rules typically depends on the size of knowledge base, which can be very slow in practice.
Grounding: Markov Logic Network

R1 2.0 \( \forall X, Y \ \text{links}(X, Y) \lor \text{links}(Y, X) \implies \text{similar}(X, Y) \)

R2 1.5 \( \forall X, Y \ \text{similar}(X, Y) \implies (\text{aboutSports}(X) \iff \text{aboutSports}(Y)) \)

(slides from Pedro Domingos)
Problem: Markov Logic Network

- Will be $O(n^2)$ nodes in graph
- $O(n^k)$ with arity-$k$ predicates
- Graph needed to answer a query is very large
- Inference not polynomial-time in graph size

ownsStock(User,Company) ➔
#Nodes = #Users * #Companies
Let’s forget about MLN for now...
Pop Quiz!
What programming language is this???

```
about(X,Z) :- handLabeled(X,Z).
about(X,Z) :- sim(X,Y), about(Y,Z).
sim(X,Y) :- links(X,Y).
sim(X,Y) :-
    hasWord(X,W), hasWord(Y,W),
    linkedBy(X,Y,W).
```
Facts about Prolog

• general purpose logic programming language associated with AI and NLP from the 70s (Wikipedia)

• elegant, expressive, deterministic, and accurate...

• currently ranked 32nd in popular program. lang. (tiobe)... even more popular than scala, F#, awk.

but...
• does not learn weights from data.
• does not take features.
• does not scale.
the New ProPPR Language

about(X,Z) :- handLabeled(X,Z)  # base.
about(X,Z) :- sim(X,Y),about(Y,Z)  # prop.
sim(X,Y) :- links(X,Y)  # sim,link.
sim(X,Y) :-
    hasWord(X,W),hasWord(Y,W),
    linkedBy(X,Y,W)  # sim,word.
linkedBy(X,Y,W) :- true  # by(W).
.. and search space...

```prolog
about(X, Z) :- handLabeled(X, Z)  # base.
about(X, Z) :- sim(X, Y), about(Y, Z)  # prop.
sim(X, Y) :- links(X, Y)  # sim,link.
sim(X, Y) :-
    hasWord(X, W), hasWord(Y, W),
    linkedBy(X, Y, W)  # sim,word.
linkedBy(X, Y, W) :- true  # by(W).
```
PPR Inference

• Score for a query soln (e.g., “Z=sport” for “about(a,Z)”) depends on *probability* of reaching a □ node
  • implicit “reset” transitions with \((p \geq \alpha)\) back to query node
• Looking for answers supported by *many short proofs*

“Grounding” size is \(O(1/\alpha \varepsilon)\) ... ie *independent* of DB size \(\rightarrow\) fast approx incremental inference (Andersen, Chung, Lang 08)
Supervised PPR Learning

• Goal: learn transition probabilities based on features of the rules.
• Backstrom & Leskovec 2011: L-BFGS with WMW loss.
• Our approach:
  • epoch-based SGD with L2-regularized log loss.
  • easy to implement.
  • single pass (fast).
  • cheap.
  • disk-friendly.
Entity Resolution

• Task:
  • citation matching (Alchemy: Poon & Domingos).

• Dataset:
  • CORA dataset, 1295 citations of 132 distinct papers.

• Training set: section 1-4.
• Test set: section 5.

• ProPPR program:
  • translated from corresponding Markov logic network (dropping non-Horn clauses)

• # of rules: 21.
# ProPPR for Entity Resolution

Table 4: ProPPR program used for entity resolution.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>samebib(BC1,BC2) :- author(BC1,A1),sameauthor(A1,A2),authorinverse(A2,BC2)</td>
<td># author.</td>
</tr>
<tr>
<td>samebib(BC1,BC2) :- title(BC1,A1),sametitle(A1,A2),titleinverse(A2,BC2)</td>
<td># title.</td>
</tr>
<tr>
<td>samebib(BC1,BC2) :- venue(BC1,A1),samevenue(A1,A2),venueinverse(A2,BC2)</td>
<td># venue.</td>
</tr>
<tr>
<td>samebib(BC1,BC2) :- sametitle(BC1,BC3),samebib(BC3,BC2)</td>
<td># tcbib.</td>
</tr>
<tr>
<td>sameauthor(A1,A2) :- haswordauthor(A1,W),haswordauthorinverse(W,A2),keyauthorword(W)</td>
<td># authorword.</td>
</tr>
<tr>
<td>sameauthor(A1,A2) :- sameauthor(A1,A3),sameauthor(A3,A2)</td>
<td># tcauthor.</td>
</tr>
<tr>
<td>sametitle(A1,A2) :- haswordtitle(A1,W),haswordtitleinverse(W,A2),keytitleword(W)</td>
<td># titleword.</td>
</tr>
<tr>
<td>sametitle(A1,A2) :- sametitle(A1,A3),sametitle(A3,A2)</td>
<td># tcTitle.</td>
</tr>
<tr>
<td>samevenue(A1,A2) :- haswordvenue(A1,W),haswordvenueinverse(W,A2),keyvenueword(W)</td>
<td># venueword.</td>
</tr>
<tr>
<td>samevenue(A1,A2) :- samevenue(A1,A3),samevenue(A3,A2)</td>
<td># tcVenue.</td>
</tr>
<tr>
<td>keyauthorword(W) :- true</td>
<td># authorWord(W).</td>
</tr>
<tr>
<td>keytitleword(W) :- true</td>
<td># titleWord(W).</td>
</tr>
<tr>
<td>keyvenueword(W) :- true</td>
<td># venueWord(W).</td>
</tr>
</tbody>
</table>
Inference Time: Citation Matching vs MLN (Alchemy)

“Grounding” is independent of DB size
## AUC: Citation Matching

<table>
<thead>
<tr>
<th></th>
<th>Cites</th>
<th>Authors</th>
<th>Venues</th>
<th>Titles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MLN</strong> Our rules</td>
<td>0.513</td>
<td>0.532</td>
<td>0.602</td>
<td>0.544</td>
</tr>
<tr>
<td><strong>ProPPR</strong>(w=1)</td>
<td>0.680</td>
<td>0.836</td>
<td>0.860</td>
<td>0.908</td>
</tr>
<tr>
<td><strong>ProPPR</strong></td>
<td>0.800</td>
<td>0.840</td>
<td>0.869</td>
<td>0.900</td>
</tr>
</tbody>
</table>

AUC scores: 0.0=low, 1.0=hi
w=1 is before learning
Learning can be parallelized

- **Learning** uses many example queries
  - e.g: sameCitation(c120,X) with X=c123+, X=c124-, ...

- Each query is grounded to a separate **small graph** (for its proof)

- Goal is to **tune weights** on these edge features to optimize RWR on the query-graphs.

- Can do **SGD** and run RWR **separately** on each query-graph
  - Graphs do share edge features, so there’s some synchronization needed
Reason on Large Knowledge Graphs

**PRA**: learning *inference* rules for a noisy KB
(Lao, Cohen, Mitchell 2011)  
(Lao et al, 2012)

- Paths are learned separately for each relation type, and one learned rule can’t call another
- PRA can only learn from facts in KB.

\[
\text{athletePlaySport} \text{ViaRule}(\text{Athlete,Sport}) : - \\
\text{onTeam} \text{ViaKB}(\text{Athlete,Team}), \text{teamPlaysSport} \text{ViaKB}(\text{Team,Sport})
\]

\[
\text{teamPlaysSport} \text{ViaRule}(\text{Team,Sport}) : - \\
\text{memberOf} \text{ViaKB}(\text{Team,Conference}), \text{hasMember} \text{ViaKB}(\text{Conference,Team2}), \text{plays} \text{ViaKB}(\text{Team2,Sport}).
\]

\[
\text{teamPlaysSport} \text{ViaRule}(\text{Team,Sport}) : - \\
\text{onTeam} \text{ViaKB}(\text{Athlete,Team}), \text{athletePlaysSport} \text{ViaKB}(\text{Athlete,Sport})
\]
Joint Inference ProPPR program

athletePlaySport(Athlete, Sport) :-
  fact_athletePlaySport(Athlete, Sport).

athletePlaySport(Athlete, Sport) :-
onTeam(Athlete, Team), teamPlaysSport(Team, Sport).

teamPlaysSport(Team, Sport) :-
  member(Team, Conference),
  member(Team2, Conference),
  plays(Team2, Sport).

teamPlaysSport(Team, Sport) :-
onTeam(Athlete, Team), athletePlaysSport(Athlete, Sport).

non-recursive rules.

recursive rules.
Joint Inference for Relation Prediction

- Train on NELL’s KB as of iteration 713
- Test on new facts from later iterations
- Try three “subdomains” of NELL
  - pick a seed entity S
  - pick top M entities nodes in a (simple untyped RWR) from S
  - project KB to just these M entities
  - look at three subdomains, six values of M
## Joint Inference

<table>
<thead>
<tr>
<th>Dataset-Model</th>
<th>Baseball</th>
<th>Google</th>
<th>Beatles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1K NR</td>
<td>0.8958</td>
<td>0.8490</td>
<td>0.7593</td>
</tr>
<tr>
<td>Top-1K R</td>
<td>0.9982</td>
<td>0.9668</td>
<td>0.8136</td>
</tr>
<tr>
<td>Top-2K NR</td>
<td>0.9193</td>
<td>0.8358</td>
<td>0.8520</td>
</tr>
<tr>
<td>Top-2K R</td>
<td>0.9998</td>
<td>0.9958</td>
<td>0.9940</td>
</tr>
<tr>
<td>Top-5K NR</td>
<td>0.8528</td>
<td>0.7750</td>
<td>0.8243</td>
</tr>
<tr>
<td>Top-5K R</td>
<td>0.9993</td>
<td>0.9962</td>
<td>0.9973</td>
</tr>
<tr>
<td>Top-10K NR</td>
<td>0.7503</td>
<td>0.7733</td>
<td>0.8136</td>
</tr>
<tr>
<td>Top-10K R</td>
<td>0.9903</td>
<td>0.9914</td>
<td>0.9973</td>
</tr>
<tr>
<td>Top-20K NR</td>
<td>0.7646</td>
<td>0.7538</td>
<td>0.7207</td>
</tr>
<tr>
<td>Top-20K R</td>
<td>0.9891</td>
<td>0.9871</td>
<td>0.9861</td>
</tr>
<tr>
<td>Top-30K NR</td>
<td>0.7746</td>
<td>0.7745</td>
<td>0.7616</td>
</tr>
<tr>
<td>Top-30K R</td>
<td>0.9892</td>
<td>0.9892</td>
<td>0.9886</td>
</tr>
</tbody>
</table>
ProPPR vs Alchemy

• Alchemy takes >4 days to train discriminatively on recursive theory with 500-entity sample
• Alchemy’s pseudo-likelihood training fails on some recursive rule sets
More with ProPPR

\[ c_1: \text{predictedClass}(\text{Doc}, Y) :- \]
\[ \text{possibleClass}(Y), \]
\[ \text{hasWord}(\text{Doc}, W), \]
\[ \text{related}(W, Y) \neq c_1. \]
\[ c_2: \text{related}(W, Y) :- \text{true} \]
\[ \# \text{relatedFeature}(W, Y) \]

\[ c_3: \text{predictedClass}(\text{Doc}, Y) :- \]
\[ \text{similar}(\text{Doc}, \text{OtherDoc}), \]
\[ \text{predictedClass}(\text{OtherDoc}, Y) \neq c_3. \]
\[ c_4: \text{similar}(\text{Doc}_1, \text{Doc}_2) :- \]
\[ \text{hasWord}(\text{Doc}_1, W), \]
\[ \text{inDoc}(W, \text{Doc}_2) \neq c_4. \]
\[ c_5: \text{predictedClass}(\text{Doc}, Y) :- \]
\[ \text{previous}(\text{Doc}, \text{OtherDoc}), \]
\[ \text{predictedClass}(\text{OtherDoc}, \text{OtherY}), \]
\[ \text{transition}(\text{OtherY}, Y) \neq c_5. \]
\[ c_6: \text{transition}(Y_1, Y_2) :- \text{true} \]
\[ \# \text{transitionFeature}(Y_1, Y_2) \]

- \( C_1 + C_2 = \) bag-of-words classifier.
- \( C_1 + C_2 + C_3 + C_4 = \) label propagation.
- \( C_1 + C_2 + C_5 + C_6 = \) HMM-like sequence classifier.
Conclusions

• We proposed a new **probabilistic programming language** that combines logical forms and graphical modeling.

• Our method is highly **scalable**, and learning can be **parallelized**.

• We obtained **promising** results in some sample tasks, including a joint relation inference task.
Thank You & Happy Halloween!

http://www.cs.cmu.edu/~yww

ww@cmu.edu