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

MACHINE LEARNING DEPARTMENT

Carnegie Mellon University

The Problem

- Task: learning to reason on large graphs.
- Approach: 1st-order probabilistic logic inference.

Origin(William, Y)

Friend(P1, P2) , Origin(P2, Y) => Origin(P1, Y)

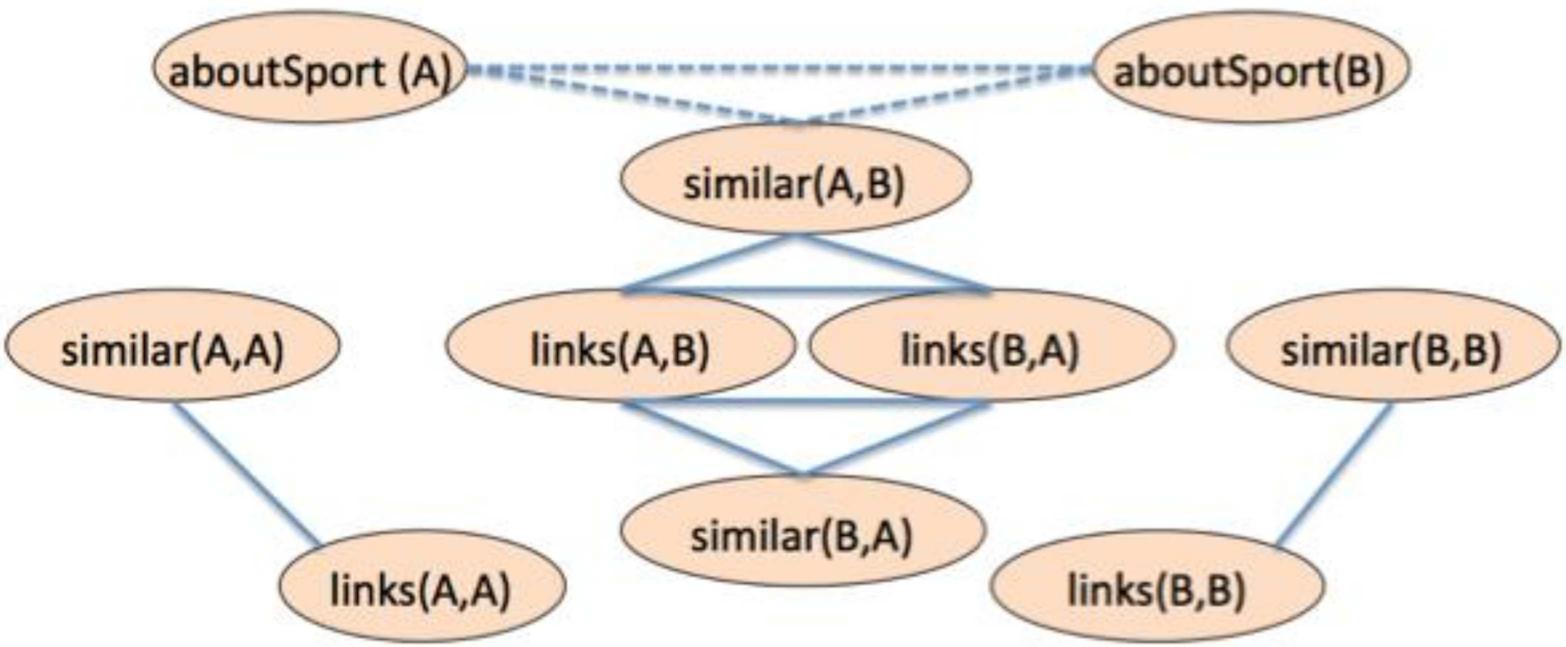
Motivation

- 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 $\forall X,Y\ links(X,Y) \vee links(Y,X) \Rightarrow similar(X,Y)$

R2 1.5 $\forall X,Y\ similar(X,Y) \Rightarrow (aboutSports(X) \Leftrightarrow aboutSports(Y))$

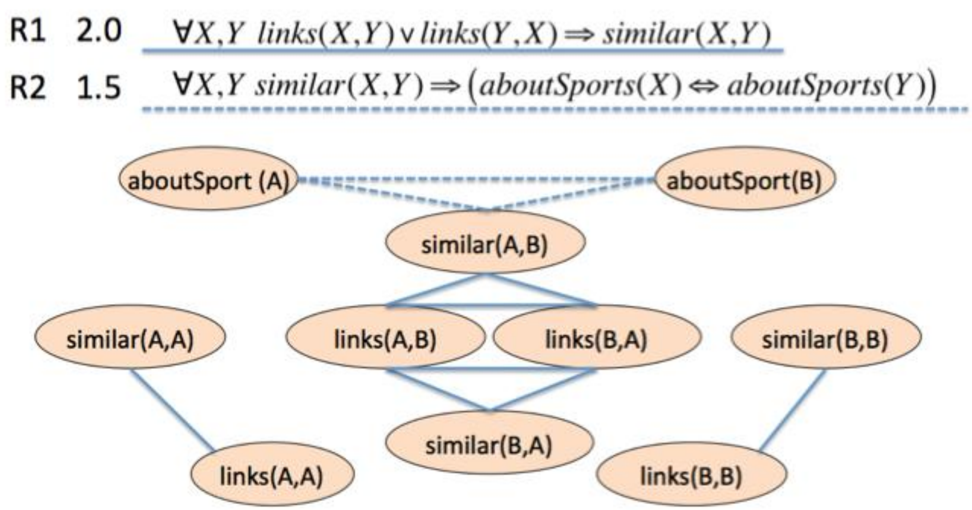


(slides from Pedro Domingos)

Carnegie Mellon
SCHOOL OF COMPUTER SCIENCE

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…

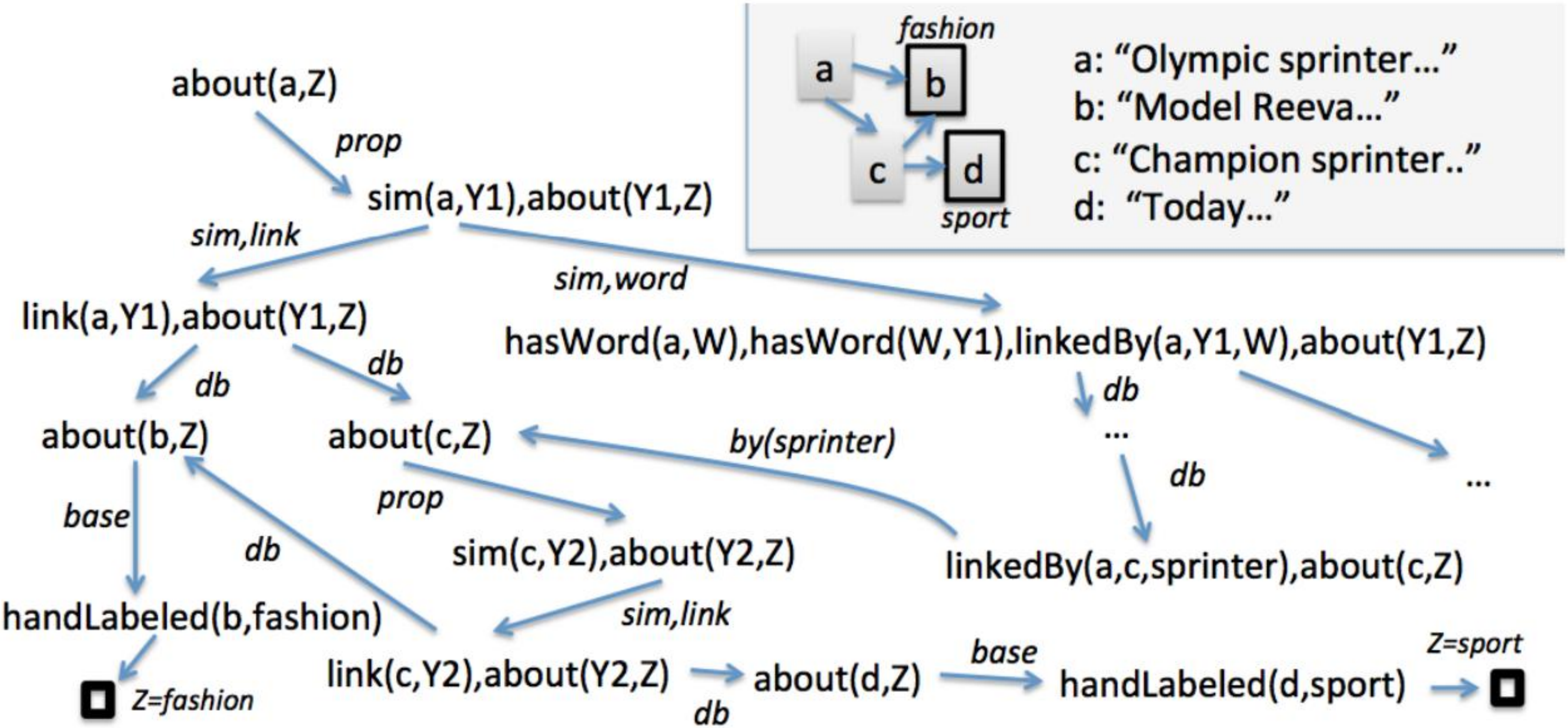
- does not learn weights from data.
- does not take features.
- does not scale.

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).
```

rules

features of rules



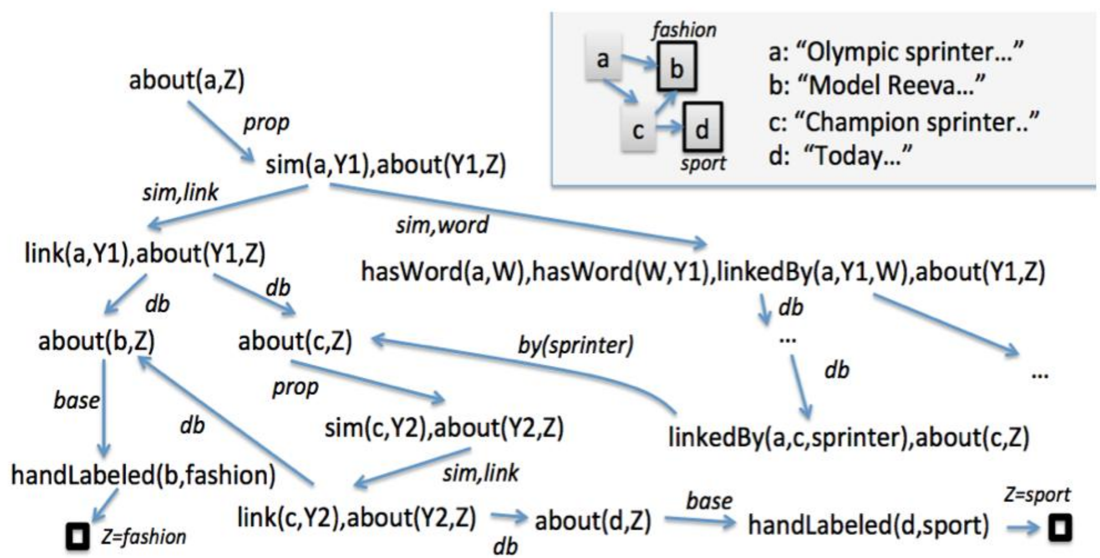
.. and search space…

```
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

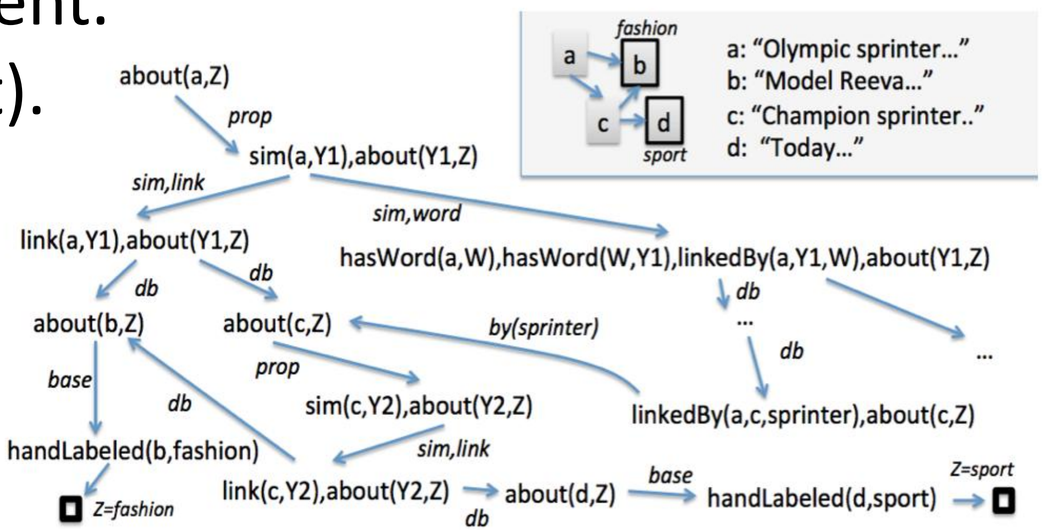
- Score for a query soln (e.g., “Z=sport” for “about(a,Z)”) depends on *probability* of reaching a ☐ node)
  - implicit “**reset**” transitions with (p≥**α**) back to query node
- Looking for answers supported by *many short proofs*

“Grounding” size is O(1/αε) … ie *independent* of DB size ➔ 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.

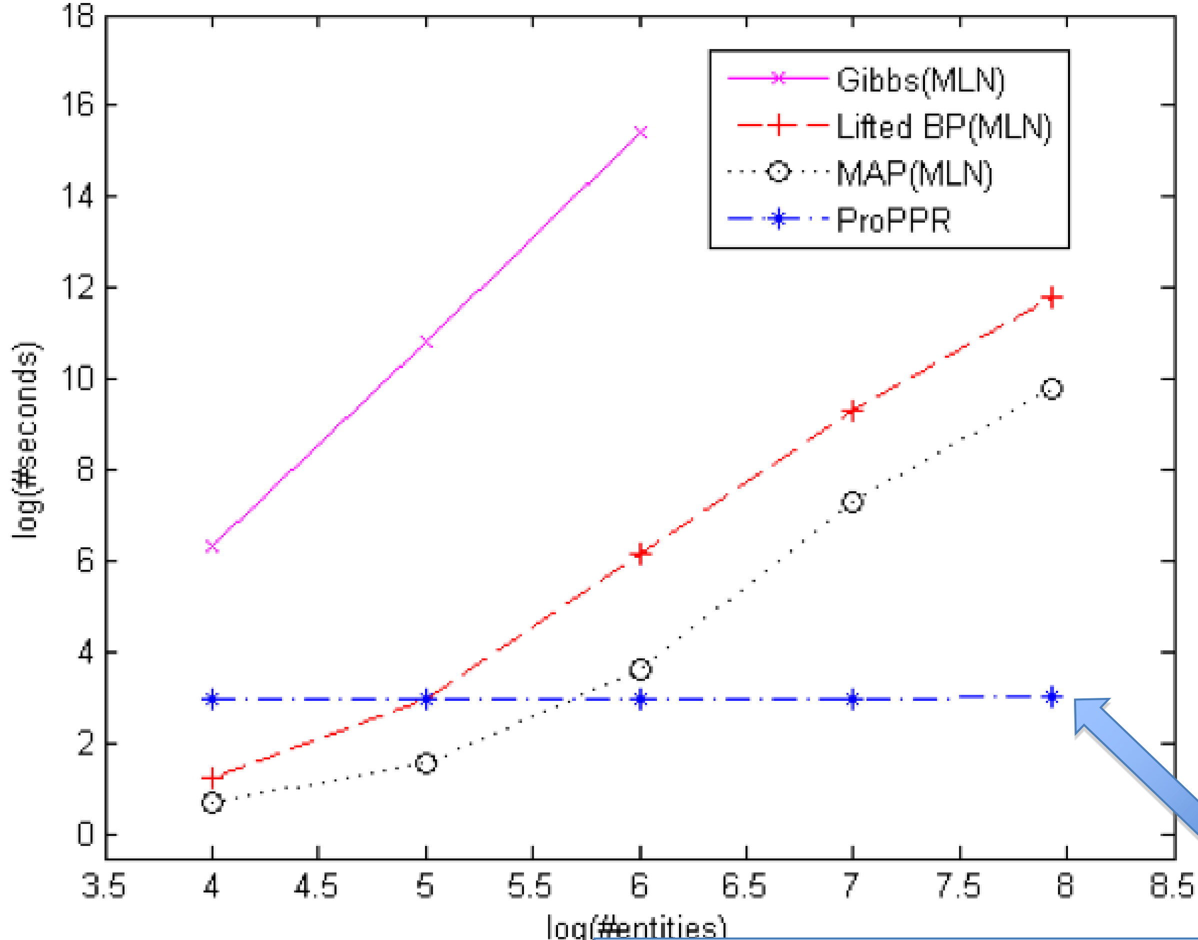
ProPPR for Entity Resolution

Table 4: ProPPR program used for entity resolution.

samebib(BC1,BC2) :- author(BC1,A1),sameauthor(A1,A2),authorinverse(A2,BC2)	# author.
samebib(BC1,BC2) :- title(BC1,A1),sametitle(A1,A2),titleinverse(A2,BC2)	# title.
samebib(BC1,BC2) :- venue(BC1,A1),samevenue(A1,A2),venueinverse(A2,BC2)	# venue.
samebib(BC1,BC2) :- samebib(BC1,BC3),samebib(BC3,BC2)	# tcbib.
sameauthor(A1,A2) :- haswordauthor(A1,W),haswordauthorinverse(W,A2),keyauthorword(W)	# authorword.
sameauthor(A1,A2) :- sameauthor(A1,A3),sameauthor(A3,A2)	# tcauthor.
sametitle(A1,A2) :- haswordtitle(A1,W),haswordtitleinverse(W,A2),keytitleword(W)	# titleword.
sametitle(A1,A2) :- sametitle(A1,A3),sametitle(A3,A2)	# tctitle.
samevenue(A1,A2) :- haswordvenue(A1,W),haswordvenueinverse(W,A2),keyvenueword(W)	# venueword.
samevenue(A1,A2) :- samevenue(A1,A3),samevenue(A3,A2)	# tcvenue.
keyauthorword(W) :- true	# authorWord(W).
keytitleword(W) :- true	# titleWord(W).
keyvenueword(W) :- true	# venueWord(W).

Inference Time: Citation Matching

vs MLN (Alchemy)



“Grounding” is independent of DB size

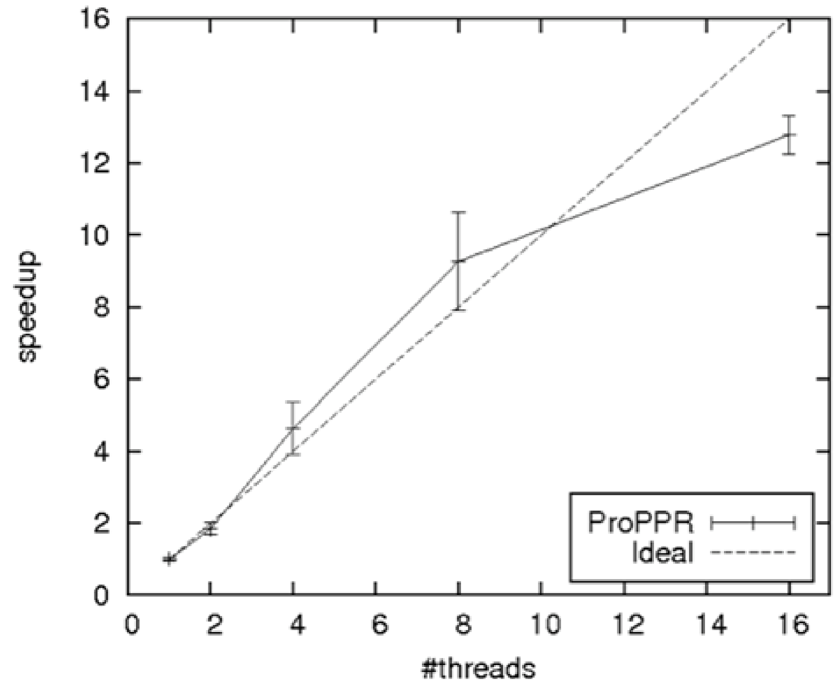
AUC: Citation Matching

	Cites	Authors	Venues	Titles
MLN Our rules	0.513	0.532	0.602	0.544
ProPPR($\mathbf{w}$=1)	0.680	0.836	0.860	**0.908**
ProPPR	**0.800**	**0.840**	**0.869**	0.900

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.

```
athletePlaySportViaRule(Athlete,Sport) :-
    onTeamViaKB(Athlete,Team), teamPlaysSportViaKB(Team,Sport)

teamPlaysSportViaRule(Team,Sport) :-
    memberOfViaKB(Team,Conference),
    hasMemberViaKB(Conference,Team2),
    playsViaKB(Team2,Sport).
teamPlaysSportViaRule(Team,Sport) :-
    onTeamViaKB(Athlete,Team), athletePlaysSportViaKB(Athlete,Sport)
```

Joint Inference ProPPR program

$$athletePlaySport(Athlete, Sport) :- fact\_athletePlaySport(Athlete, Sport).$$

non-recursive rules.

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

recursive rules.

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

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

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

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

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: predictedClass(Doc,Y) :-
possibleClass(Y),
hasWord(Doc,W),
related(W,Y) # c1.
c_2: related(W,Y) :- true
relatedFeature(W,Y)

Database predicates:
hasWord(D,W): doc D contains word W
inDoc(W,D): doc D contains word W
previous(D1,D2): doc D2 precedes D1
possibleClass(Y): Y is a class label

c_3: predictedClass(Doc,Y) :-
similar(Doc,OtherDoc),
predictedClass(OtherDoc,Y) # c3.
c_4 : similar(Doc1,Doc2) :-
hasWord(Doc1,W),
inDoc(W,Doc2) # c4.

c_5 : predictedClass(Doc,Y) :-
previous(Doc,OtherDoc),
predictedClass(OtherDoc,OtherY),
transition(OtherY,Y) # c5.
c_6: transition(Y1,Y2) :- true
transitionFeature(Y1,Y2)

- C1 + C2 = bag-of-words classifier.
- C1 + C2 + C3 + C4 = label propagation.
- C1 + C2 + C5 + C6 = 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