Random Walk Inference and Learning in A Large Scale Knowledge Base

Anshul Bawa

Adapted from slides by:
Ni Lao, Tom Mitchell, William W. Cohen

6 March 2017
Outline

- Inference in Knowledge Bases
- The NELL project and N-FOIL
- Random Walk Inference : PRA
- Task formulation
- Heuristics and sampling
- Evaluation
- Class discussion
Challenges to Inference in KBs

- Traditional logical inference methods too brittle - Robustness
- Probabilistic inference methods not scalable - Scalability
NELL

Combines multiple strategies:
- morphological patterns
- textual context
- html patterns
- logical inference

Half million confident beliefs
Several million candidate beliefs
Horn Clause Inference

N-FOIL algorithm:
  ● start with a general rule
  ● progressively specialize it
  ● learn a clause
  ● remove examples covered

\[
\text{AthletePlaysForTeam}(x, y) \\
\land \quad \text{TeamPlaysInLeague}(y, z) \\
\Rightarrow \quad \text{AthletePlaysInLeague}(x, z)
\]

Computationally expensive
Horn Clause Inference

Assumptions :

- Functional predicates only : No need for negative examples

- Relational pathfinding : Only clauses from bounded paths of binary relations

Small no. (~600) of high precision rules
Horn Clause Inference

Issues:

- Still costly: N-FOIL takes days on NELL
- Combination by disjunction only: cannot leverage low-accuracy rules
Horn Clause Inference

Issues:

- Still costly: N-FOIL takes days on NELL
- Combination by disjunction only: cannot leverage low-accuracy rules
- High precision but low recall
Random Walks Inference

Labeled, directed graph

- each entity $x$ is a node
- each binary relation $R(x,y)$ is an edge labeled $R$ between $x$ and $y$
- unary concepts $C(x)$ are represented as edge labeled “isa” between the node for $x$ and a node for the concept $C$
Random Walks Inference

Labeled, directed graph
- each entity $x$ is a node
- each binary relation $R(x,y)$ is an edge labeled $R$ between $x$ and $y$
- unary concepts $C(x)$ are represented as edge labeled "isa" between the node for $x$ and a node for the concept $C$

Given a node $x$ and relation $R$, give ranked list of $y$
Random Walks Inference: PRA

Logistic Regression over a large set of experts

Each expert is a bounded-length sequence of edge-labeled path types

\[
\text{isa}(x, c) \land \text{isa}^{-1}(c, x') \\
\land \quad \text{AthletePlaysInLeague}(x', y) \\
\Rightarrow \quad \text{AthletePlaysInLeague}(x, y)
\]
Random Walks Inference:

PRA

Logistic Regression over a large set of experts
Each expert is a bounded-length sequence of edge-labeled path types
Expert scores are relational features
Score(y) = |A_y| / |A|
Many such low-precision high-recall experts
Random Walks Inference: PRA

Logistic Regression over a large set of experts

Each expert is a bounded-length sequence of edge-labeled path types

Expert scores are relational features

Score(y) = \frac{|A_y|}{|A|}

Many such low-precision high-recall experts

\[
isa(x, c) \land isa^{-1}(c, x') \\
\land AthletePlaysInLeague(x', y) \\
\Rightarrow AthletePlaysInLeague(x, y)
\]
Path Ranking Algorithm

[Lao & Cohen ECML2010]

A relation path $P=(R_1, \ldots, R_n)$ is a sequence of relations.

A PRA model scores a source-target node pair by a linear function of their path features.

$$\text{score}(e; s) = \sum_{P \in \mathbf{P}_\ell} h_{s, P}(e) \theta_P$$

$$h_{s, P}(e) = \sum_{e' \in \text{range}(P')} h_{s, P'}(e') \cdot P(e|e'; R_\ell)$$

$$h_{s, P}(e) = \begin{cases} 1, & \text{if } e = s \\
0, & \text{otherwise} \end{cases}$$
Path Ranking Algorithm

[Laó&Cohen ECML2010]

Training:

For a relation $R$ and a set of node pairs $\{(s_i, t_i)\}$, we construct a training dataset $D = \{(x_i, y_i)\}$, where

- $x_i$ is a vector of all the path features for $(s_i, t_i)$, and
- $y_i$ indicates whether $R(s_i, t_i)$ is true
- $\theta$ is estimated using regularized logistic regression
Link Prediction Task

Consider 48 relations for which NELL database has more than 100 instances.

Two link prediction tasks for each relation:
- AthletePlaysInLeague(HinesWard, ?)
- AthletePlaysInLeague(?, NFL)

The actual nodes $y$ known to satisfy $R(x; ?)$ are treated as labeled positive examples, and all other nodes are treated as negative examples.
Captured paths/rules

Broad coverage rules

**athletePlaysSport**

\[ c \xRightarrow{\text{isa}} c \xRightarrow{\text{isa}^{-1}} c \xRightarrow{\text{athletePlaysSport}} c \text{ (popular sports of all the athletes)} \]

\[ c \xRightarrow{\text{athletePlaysInLeague}} c \xRightarrow{\text{superpartOfOrganization}} c \xRightarrow{\text{teamPlaysSport}} c \text{ (popular sports of a certain league)} \]

Accurate rules

**teamPlaysInCity**

\[ c \xRightarrow{\text{teamHomeStadium}} c \xRightarrow{\text{stadiumLocatedInCity}} c \text{ (city of the team’s home stadium)} \]
Captured paths/rules

Broad coverage rules

\[ \text{athletePlaysSport} \]
\[
\begin{array}{c}
\text{isa} \\
\text{isa}^{-1}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\rightarrow
\begin{array}{c}
\text{athletePlaysSport}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\quad \text{(popular sports of all the athletes)}
\]

\[
\begin{array}{c}
\text{isa}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\rightarrow
\begin{array}{c}
\text{athletePlaysInLeague}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\rightarrow
\begin{array}{c}
\text{superpartOfOrganization}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\rightarrow
\begin{array}{c}
\text{teamPlaysSport}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\quad \text{(popular sports of a certain league)}
\]

Accurate rules

\[ \text{teamPlaysInCity} \]
\[
\begin{array}{c}
\text{teamHomeStadium}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\rightarrow
\begin{array}{c}
\text{stadiumLocatedInCity}
\end{array}
\rightarrow
\begin{array}{c}
\text{c}
\end{array}
\quad \text{(city of the team’s home stadium)}
\]
Captured paths/rules

Rules with synonym information

\[
\text{teamMember} \xrightarrow{C} \text{athletePlaysForTeam} \xrightarrow{C} \text{teamHomeStadium} \xrightarrow{C}
\]

(home stadium of teams which share players with the query)

Rules with neighbourhood information

\[
\text{teamPlaysInLeague} \xrightarrow{C} \text{teamPlaysSport} \xrightarrow{C} \text{players} \xrightarrow{C} \text{athletePlaysInLeague} \xrightarrow{C}
\]
Captured paths/rules

Rules with synonym information

\[\text{teamMember} \xrightarrow{C} \text{athletePlaysForTeam} \xrightarrow{C} \text{teamHomeStadium} \xrightarrow{C} (\text{home stadium of teams which share players with the query})\]

Rules with neighbourhood information

\[\text{teamPlaysInLeague} \xrightarrow{C} \text{teamPlaysSport} \xrightarrow{C} \text{players} \xrightarrow{C} \text{athletePlaysInLeague} \xrightarrow{C}\]
Data-driven Path finding

Impractical to enumerate all possible paths, even for small length $l$

- Require any path to instantiate in at least $\alpha$ portion of the training queries, i.e. $h_{s,P}(t) \neq 0$ for any $t$
- Require any path to reach at least one target node in the training set
Data-driven Path finding

Impractical to enumerate all possible paths, even for small length $l$

- Require any path to instantiate in at least $\alpha$ portion of the training queries, i.e. $h_{s,p}(t) \neq 0$ for any $t$
- Require any path to reach at least one target node in the training set

Discover paths by a depth first search: Starts from a set of training queries, expand a node if the instantiation constraint is satisfied
Data-driven Path finding

Discover paths by a depth first search: Starts from a set of training queries, expand a node if the instantiation constraint is satisfied

Dramatically reduce the number of paths

Table 1: Number of paths in PRA models of maximum path length 3 and 4. Averaged over 96 tasks.

<table>
<thead>
<tr>
<th>Condition</th>
<th>$\ell=3$</th>
<th>$\ell=4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>all paths up to length $L$</td>
<td>15,376</td>
<td>1,906,624</td>
</tr>
<tr>
<td>+query support $\geq \alpha = 0.01$</td>
<td>522</td>
<td>5016</td>
</tr>
<tr>
<td>+ever reach a target entity</td>
<td>136</td>
<td>792</td>
</tr>
<tr>
<td>+$L_1$ regularization</td>
<td>63</td>
<td>271</td>
</tr>
</tbody>
</table>
Low-Variance Sampling

[Laos& Cohen KDD2010]

Exact calculation of random walk distributions results in non-zero probabilities for many internal nodes.

But computation should be focused on the few target nodes which we care about.
Low-Variance Sampling
[Lao & Cohen KDD 2010]

A few random walkers (or particles) are enough to distinguish good target nodes from bad ones.

Sampling walkers/particles independently introduces variances to the result distributions.
Low-Variance Sampling

Instead of generating independent samples from a distribution, LVS uses a single random number to generate all samples.

Given a distribution $P(x)$, any number $r$ in $[0, 1]$ corresponds to exactly one $x$ value, namely

$$x = \arg \min_j \sum_{m=1..j} P(m) \leq r$$
Low-Variance Sampling

Instead of generating independent samples from a distribution, LVS uses a single random number to generate all samples

Given a distribution \( P(x) \), any number \( r \) in \([0, 1]\) corresponds to exactly one \( x \) value, namely

\[
x = \arg \min_j \sum_{m=1..j} P(m) \leq r
\]

To generate \( M \) samples from \( P(x) \), generate a random \( r \) in the interval \([0, 1/M]\)

Repeatedly add the fixed amount \( 1/M \) to \( r \) and choose \( x \) values corresponding to the resulting numbers
Low-Variance Sampling

Instead of generating independent samples from a distribution, LVS uses a single random number to generate all samples.

Given a distribution \( P(x) \), any number \( r \) in \([0, 1]\) corresponds to exactly one \( x \) value, namely

\[
x = \arg \min_j \sum_{m=1..j} P(m) \leq r
\]

To generate \( M \) samples from \( P(x) \), generate a random \( r \) in the interval \([0, 1/M]\).

Repeatedly add the fixed amount \( 1/M \) to \( r \) and choose \( x \) values corresponding to the resulting numbers.
Comparison

Inductive logic programming (e.g. FOIL)
- Brittle facing uncertainty

Statistical relational learning (e.g. MLNs, Relational Bayesian Networks)
- Inference is costly when the domain contains many nodes
- Inference is needed at each iteration of optimization

Random walk inference
- Decouples feature generation and learning: No inference during optimization
- Sampling schemes for efficient random walks: Trains in minutes, not days
- Low precision/high recall rules as features with fractional values: Doubles precision at rank 100 compared with N-FOIL
- Handles non-functional predicates
Comparison

Inductive logic programming (e.g. FOIL)
- Brittle facing uncertainty

Statistical relational learning (e.g. MLNs, Relational Bayesian Networks)
- Inference is costly when the domain contains many nodes
- Inference is needed at each iteration of optimization

Random walk inference
- Decouples feature generation and learning: No inference during optimization
- Sampling schemes for efficient random walks: Trains in minutes, not days
- Low precision/high recall rules as features with fractional values: Doubles precision at rank 100 compared with N-FOIL
- Handles non-functional predicates

+ Dinesh + Rishal + Barun + Nupur + Arindam + Shantanu + Surag
Eval : Cross-val on training

Table 2: Comparing PRA with RWR models. MRRs and training times are averaged over 96 tasks.

<table>
<thead>
<tr>
<th></th>
<th>$\ell=2$</th>
<th></th>
<th>$\ell=3$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>Time</td>
<td>MRR</td>
<td>Time</td>
</tr>
<tr>
<td>RWR(no train)</td>
<td>0.271</td>
<td></td>
<td>0.456</td>
<td></td>
</tr>
<tr>
<td>RWR</td>
<td>0.280</td>
<td>3.7s</td>
<td>0.471</td>
<td>9.2s</td>
</tr>
<tr>
<td>PRA</td>
<td>0.307</td>
<td>5.7s</td>
<td>0.516</td>
<td>15.4s</td>
</tr>
</tbody>
</table>

Mean Reciprocal Rank : inverse rank of the highest ranked relevant result (higher is better)
Eval : Cross-val on training

Table 2: Comparing PRA with RWR models. MRRs and training times are averaged over 96 tasks.

<table>
<thead>
<tr>
<th></th>
<th>$\ell=2$</th>
<th></th>
<th>$\ell=3$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>Time</td>
<td>MRR</td>
<td>Time</td>
</tr>
<tr>
<td>RWR(no train)</td>
<td>0.271</td>
<td></td>
<td>0.456</td>
<td></td>
</tr>
<tr>
<td>RWR</td>
<td>0.280</td>
<td>3.7s</td>
<td>0.471</td>
<td>9.2s</td>
</tr>
<tr>
<td>PRA</td>
<td>0.307</td>
<td>5.7s</td>
<td>0.516</td>
<td>15.4s</td>
</tr>
</tbody>
</table>

Mean Reciprocal Rank : inverse rank of the highest ranked relevant result (higher is better)
Eval : Cross-val on training

Table 2: Comparing PRA with RWR models. MRRs and training times are averaged over 96 tasks.

<table>
<thead>
<tr>
<th></th>
<th>$\ell=2$ MRR</th>
<th>$\ell=2$ Time</th>
<th>$\ell=3$ MRR</th>
<th>$\ell=3$ Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWR(no train)</td>
<td>0.271</td>
<td></td>
<td>0.456</td>
<td></td>
</tr>
<tr>
<td>RWR</td>
<td>0.280</td>
<td>3.7s</td>
<td>0.471</td>
<td>9.2s</td>
</tr>
<tr>
<td>PRA</td>
<td>0.307</td>
<td>5.7s</td>
<td>0.516</td>
<td>15.4s</td>
</tr>
</tbody>
</table>

Supervised training can improve retrieval quality (RWR)

RWR : One-parameter-per-edge label, ignores context

Path structure can produce further improvement (PRA)
Eval : Effect of sampling

LVS can slightly improve prediction for both finger printing and particle filtering.

Figure 2: Compare inference speed and quality over 96 tasks. The speedup is relative to exact inference, which is on average 23ms per query.
AMT evaluation

<table>
<thead>
<tr>
<th>Functional Predicates</th>
<th>Non-functional Predicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Rules</td>
<td>p@10</td>
</tr>
<tr>
<td>N-FOIL 2.1(+37)</td>
<td>0.76</td>
</tr>
<tr>
<td>PRA 43</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Sorted the queries for each predicate according to the scores of their top-ranked results, and then evaluated precisions at top 10, 100 and 1000 queries.
Discussion

Dinesh: miss out on knowledge not present in the path. one hop neighbours as features?

Gagan: compare average values for highest ranked relevant result instead of MRR; comparison to MLNs

Rishab, Barun, Surag: Analysis of low MRR/errors

Rishab: low path scores for more central nodes

Shantanu: ignoring a relation in inferring itself? Same relation with different arguments
Extensions

- Multi-concept inference: Gagan
- SVM classifiers: Rishab, Nupur, Surag
- Joint inference: Paper, Rishab, Gagan, Barun, Haroun
- Relation embeddings: Rishab
- Path pruning using horn clauses: Barun
- Target node statistics: Paper, Barun, Nupur
- Tree kernel SVM: Akshay
Extensions

- Longer paths: Paper, Haroun
- Lexicalized paths: Paper
- Generalize paths to trees: Paper, Haroun
- No path between source and target; relation similarity: Arindam
- Information sharing across relations; MLN layer: Ankit
- Weigh edges by confidence: Surag
- Aligned graph on multiple data sources: Prachi
The End