ProbKB: Managing Web-Scale Knowledge

CIS6930 LRG ADV DATA ANLY

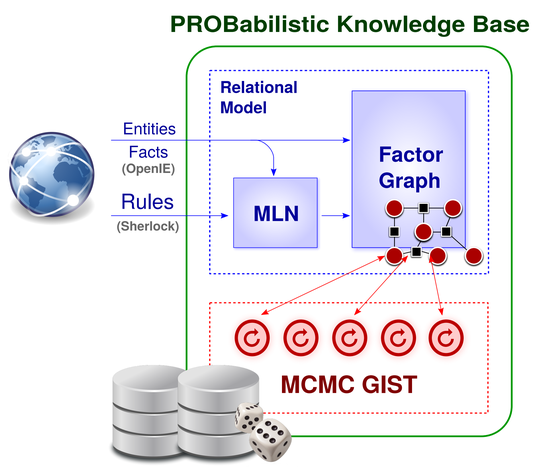
YANG CHEN[§], XING LIU[§],
[§]*CISE, University of Florida*
{yang,xinliu} @cise.ufl.edu

UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

ABSTRACT

ProbKB, a PROBabilistic Knowledge Base constructed from web scale extracted entities, facts, and rules represented as a Markov logic network (MLN). We achieved web scale MLN inference by designing a novel structured, relational model for MLNs and efficient grounding algorithms that apply rules in batches. Errors are handled in a principled and elegant manner to avoid unnecessary resource consumption. The inference task is delegated to GraphLab, a distributed framework designed specifically for machine learning problems. Our initial experiments show that our approach has much better scalability than the state-of-the-art.

SYSTEM OVERVIEW



CHALLENGES AND CURRENT WORK

- Item 1.
- Item 2.
- Item 3.
- Item 4.
- Item 5.

RELATIONAL MLN MODEL

- Put entities, facts, and rules into the database, so that the rules are applied in batches, and wrapped the grounding logic into stored procedures.
- Considered Horn clauses only.Identify six rules pattern in SHERLOCK rules.Each rule type i has a table Mi recording the predicates involved in the rules of that type.
- Another table R for relationships. For each relationship p(x, y) that is stated in the text corpus, we have a tuple (p, x, y) in R.

GROUNDING

- Used lazy inference and adopted a one-step look-ahead strategy: assume all atoms are inactive and compute active clauses. activate the atoms in the grounding result and recompute active clauses.
- Assume rules of type 3 are stored in table M3 (p, q, r), and relationships p(x, y) are stored in R(p, x, y), then the following SQL query computes atoms that are activated during the grounding process ,this process is repeated until convergence, resulting in an active closure. The following SQL query then computes active clauses given the active atoms:

```
SELECT DISTINCT
R1.id AS id1, R2.id AS id2, R3.id AS id3
FROM M3 JOIN R R ON M3.p = R.p
JOIN R R1 ON M3.q = R1.p
JOIN R R2 ON M3.r = R2.p
WHERE R.x = R1.x AND R.y = R2.x AND R1.y = R2.y
```

- The result of grounding is a factor graph (Markov network). This graph encodes a probability distribution over its variable nodes, which can be used to answer user queries. We use TUFFY as comparison point. We tried to run ProbKB using the ReVerb- Sherlock dataset and measured the grounding time in 85 seconds whereas TUFFY crashes in this step.

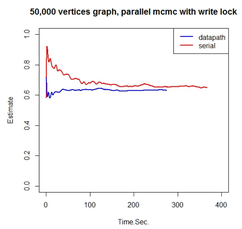
KNOWLEDGE INTEGRITY

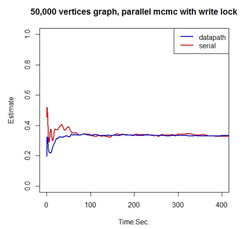
- Few common error patterns (ambiguity, meaningless relations, etc) that might affect learning quality. We use robust semi-naive evalution to emphasize the fact that inferences only occur among most correst facts and errors are unlikely to arise.

$candidates \leftarrow allfacts$
$beliefs \leftarrow promote(\varnothing, candidates)$
$delta = \varnothing$
repeat
  $promoter \leftarrow infer(beliefs, delta)$
  $beliefs \leftarrow beliefs \cup delta$
  $delta \leftarrow \varnothing$
until

INFERENCE

- Use MCMC algorithm to do the marginal inference in MLN. The results are the marginal probablity of grounding facts.
- Our first approach is using MetropolisâĂŞHastings algorithm with Datapath as our parallel engine. Partition the graph to subgraphs and runs the algorithm on each of them. We found this approach could break the Markov Logic Network if a factor is accross several subgraphs.
- Our second approach is to use a write lock on each variable on the factor graph and random walk on the whole graph instead of partition the graph, and collecting samples in parallel.
- Because of performance and accuracy issue, Our lattest approach is that runs Gibbs sampling using GraphLab as our parallel engine.
- We sampled a subset with 700 facts and compared the time spent on generating 200 joint samples for ProbKB and Tuffy. It took ProbKB 0.47 minute to complete the inference whereas TUFFY used 55 minutes.





2