Pre and Post Counting for Scalable Statistical-Relational Model Discovery

Richard Mar and Oliver Schulte

School of Computing Science

Simon Fraser University, Vancouver, Canada

Code Repository: https://github.com/sfu-cl-lab/FactorBase

Overview

- A key problem for relational learning is to *count how often a pattern occurs in the data* (e.g. pairs of women users who are friends)
- For propositional i.i.d. data, need only to filter a *subset of existing rows*
- Relational counting is much harder:

1. Need to consider *new tuples* (e.g., pairs, triples of users): the **Join Problem**
2. Need to consider non-existing links (e.g. pairs of users who are not friends): the **Negation Problem**

- Insights from graphical model learning:

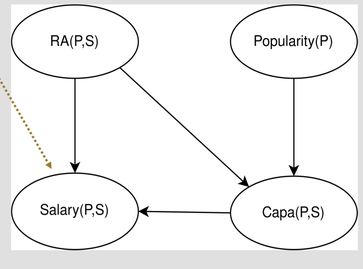
1. caching counts to avoid recounting is important for scalability
2. Important issue is when to count.
   - **Precounting** collects a large number of patterns *before* model search
   - **Ondemand** counting computes counts only for patterns generated during model search

- Conclusion: Use
  - precounting for the Join Problem
  - ondemand counting for the Negation Problem.
- This hybrid method can scale to 15M facts

Contingency Tables (CTs)

- Used for storing and transforming counts

Count	*Capa(P,S)*	*RA(P,S)*	*Salary(P,S)*
203	N/A	F	N/A
5	4	T	HIGH
4	5	T	HIGH
2	3	T	HIGH
1	3	T	LOW
2	2	T	LOW
2	1	T	LOW
2	2	T	MED
4	3	T	MED
3	1	T	MED



When to Count?

- **Precount**: compute CTs for all variables/predicates in data
- **Postcount/on-demand**: compute CTs for relational dependency (clause) only if generated during model search

Method	Level	Memory	Count time	Model Search Support
Precount P	Global	x Exponential e.g. 2^{20} for 20 predicates	x exponential ✓ dynamic programming	✓ (lookup)
On-demand O	Local	✓ near constant	✓ near constant	x generates many separate CTS

- New **Hybrid Method H:**

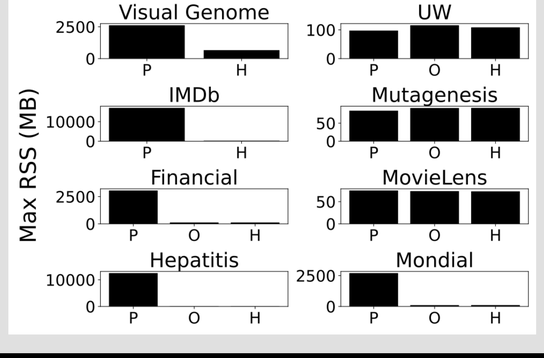
1. Precount for existing relationships (Joins)
2. On-demand for non-existing relationships (Negation)

Naïve SRL methods use on-demand without caching

Evaluation: Datasets

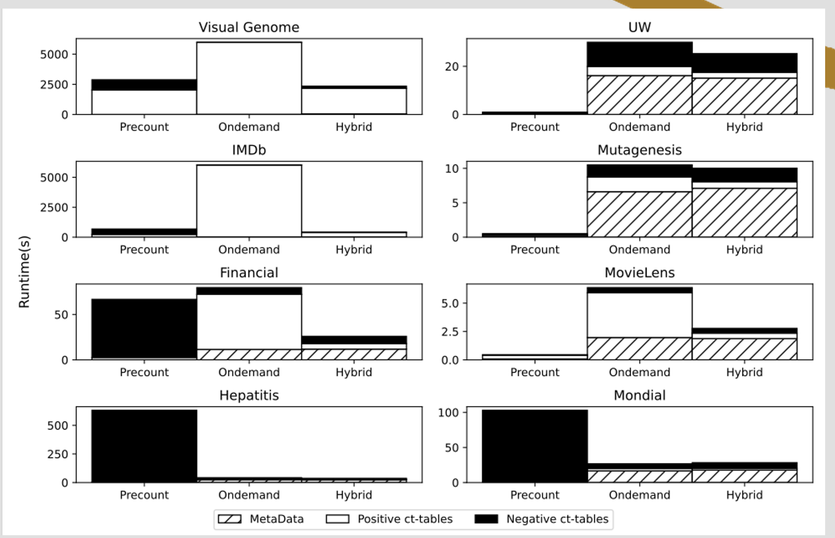
MP/N = Average BN indegree

Database	Row Count	# Relationships	MP/N
UW	712	2	1.6
Mondial	870	2	1.3
Hepatitis	12,927	3	1.7
Mutagenesis	14,540	2	1.6
MovieLens	74,402	1	1.4
Financial	225,887	3	1.9
IMDb	1,063,559	3	3.4
Visual Genome	15,833,273	8	0.5



Maximum Resident Set Size

Results



DB	Estimated $ct(family)$ Count	Total Row	$ct(database)$ Total Row Count
MovieLens	816		239
Mutagensis	6075		1631
UW	15318		2828
Visual Genome	2923968		20447
Mondial	55800		1738867
Financial	930468		3013006
Hepatitis	176220		12374892
IMDb	33040		15537457

Conclusion

- Instantiation counts are a key computational bottleneck for relational learning
- Caching is important to avoid recomputing counts
- Precounting for all possible patterns is faster than plain on-demand counting during model search
  * Assuming enough memory and moderate predicates (20 or less)
- Hybrid is even faster: precount for existing relationships, on-demand for negations

References

1. Zhensong Qian, Oliver Schulte, and Yan Sun. Computing Multi-Relational Sufficient Statistics for Large Databases. CIKM 2014, 1249–1258.
2. Lv, Q.; Xia, X. & Qian, P.
   A fast calculation of metric scores for learning Bayesian networks
   International Journal of Automation and Computing, 2012, 9, 37-44