Processing Markov Logic Networks with GPUs

Carlos Alberto Martínez-Angeles\textsuperscript{1}, Inês Dutra\textsuperscript{2}, Vítor Santos Costa\textsuperscript{2} and Jorge Buenabad-Chávez\textsuperscript{1}

\textsuperscript{1}Departamento de Computación, CINVESTAV-IPN
Av. Instituto Politécnico Nacional 2508, México D.F. 07360, México.
camartinez@cinvestav.mx, jbuenabad@cs.cinvestav.mx

\textsuperscript{2}Departamento de Ciência de Computadores, CRACS INESC-TEC LA and Universidade do Porto, Rua do Campo Alegre 1021, 4169-007, Porto, Portugal
\{ines,vsc\}@dcc.fc.up.pt

August 20-22, 2015
Contents

1. Introduction
2. Our GPU-based Markov Logic Platform
3. Results
4. Conclusions
Markov logic “combines first-order logic and Markov networks. A knowledge base in Markov logic is a set of first-order [logic] formulas with weights”

Weights establish *soft constraints*: worlds that violate a formula are less likely but still possible.

Markov logic networks have been widely adopted and are used to:

- Refine Wikipedia’s Infobox Ontology.
- Carry out collective semantic role labelling.
- Perform Natural Language Processing.

Inference is divided into a *grounding* phase and a *search* phase.
**ENGLISH AND FIRST-ORDER LOGIC** | **CLAUSAL FORM (Datalog syntax)** | **WEIGHT**
--- | --- | ---
Friends of friends are friends:  
\( \text{Fr}(x, y) \land \text{Fr}(y, z) \Rightarrow \text{Fr}(x, z) \)  
\( \text{Fr}(x, z) :- \text{Fr}(x, y), \text{Fr}(y, z) \)  
0.7
Smoking causes cancer:  
\( \text{Sm}(x) \Rightarrow \text{Ca}(x) \)  
\( \text{Ca}(x) :- \text{Sm}(x) \)  
1.5
If two people are friends and one  
smokes, then so does the other:  
\( \text{Fr}(x, y) \land \text{Sm}(x) \Rightarrow \text{Sm}(y) \)  
\( \text{Sm}(y) :- \text{Fr}(x, y), \text{Sm}(x) \)  
1.1

**EVIDENCE PROCESSING OF VALID GROUNDINGS**

| EVIDENCE | PROCESSING OF VALID GROUNDINGS | RESULTS |
--- | --- | ---
Fr(John, Anna)  
Fr(Anna, Bob)  
Fr(Gary, Frank)  
Sm(John) | Fr(John, Anna) \land Fr(Anna, Bob) \Rightarrow Fr(John, Bob)  
Sm(Anna) \Rightarrow Ca(Anna)  
Fr(Gary, Frank) \land Sm(Gary) \Rightarrow Sm(John)  
Sm(Gary) | 0.92 Ca(John)  
0.59 Ca(Anna)  
0.58 Ca(Bob) |

**PROCESSING OF INVALID GROUNDINGS**

| EVIDENCE | PROCESSING OF INVALID GROUNDINGS |
--- | ---
Fr(John, Anna) \land Fr(Gary, Frank) \Rightarrow {}  
Sm(Gary) \Rightarrow Ca(Gary)  
Fr(Gary, Frank) \land Sm(Gary) \Rightarrow Sm(Frank) |
Tuffy and other MLN systems

- Alchemy was the first MLN system. Includes various algorithms for inference and learning.
- Tuffy relies on PostgreSQL to perform the grounding using a bottom-up approach, based on SQL queries and accelerated with query optimizations by the RDBMS.
- RockIt treats inference as an integer linear programming problem. Currently outperforms all other systems.
Its main components are: Tuffy, YAP Prolog and GPU-Datalog.

GPU-Datalog evaluates Datalog programs with a bottom-up approach using GPU kernels that implement relational algebra operations.

It includes several optimizations like operator weaving, automatic memory management, among others.

It was extended with: management of negation, comparison predicates and a PostgreSQL interface.
Tu2GPU-Datalog sequence diagram II

send rules

translate rules

send rules

read tables

write results

perform grounding

finish grounding

finish grounding

read groundings

perform search

display results
Results

- Applications: Entity Resolution (ER), Relational Classification (RC) and Information Extraction (IE).
- Left plot shows performance using Tuffy’s example data. Right plot uses bigger, randomly generated data.
- Empty spaces mean that the system did not finish after several hours.
For ER, our random data and its recursive clauses generate more recursive steps, 24 vs 2 in the original data.

GPU-Datalog was designed around these recursive applications. Other systems struggled with costly joins.

For RC, the search phase alone takes an astounding 43 minutes. The rest of the process approximately 3 minutes.

During this phase, 5.5M active tuples were used. In contrast, ER uses only 252K tuples.
Our system accelerates the grounding step in MLNs by combining Tuffy with our GPU-Datalog engine.

Its performance is on par or better than other well-known MLN systems.

Results show that the benefit of performing the grounding phase on the GPU outweighs the overhead of using a database and of GPU I/O.

Our system can be greatly improved by also performing the search phase in the GPU.
Thanks