Materializing Knowledge Bases via Trigger Graphs

Efi Tsamoura
Samsung AI, Cambridge, UK
About this talk

The aim is to:

– **Part I.** Motivate the use of logic and symbolic knowledge representation and reasoning techniques in developing AI applications.

– **Part II.** Present techniques to improve the efficiency of logical inference.
Part I. Motivation
Deep learning - the successes
PReLU-nets surpass humans on classification in 2015

- PReLU networks achieve 4.94% top-5 test error on ImageNet 2012 classification.
- Human-Level top-5 test error is 5.1%.

AlphaGo seals 4-1 victory over Go grandmaster in 2016

DeepMind’s artificial intelligence astonishes fans to defeat human opponent and offers evidence computer software has mastered a major challenge.
ChatGPT passes medical and law exams in 2023

The newest version of ChatGPT passed the US medical licensing exam with flying colors — and diagnosed a 1 in 100,000 condition in seconds
Deep learning strengths

- Pattern classification (in large).
- Learning via example.
- Tolerance to noise.
Deep learning - the failures
Deep (vision) models are prone to biases

Large language models fail on abstract reasoning

- LLMs have very limited performance in abstract reasoning.
- Techniques that can improve performance on other NLP tasks cannot improve the abstract reasoning capabilities of large language models.

Deep learning weaknesses

- Focus on single cognitive abilities.
- Requires large amounts of training data.
- Lacks transparency/interpretability.
- Its answers cannot be fully trusted.
- Prone to data biases.
- Difficult to incorporate background knowledge.

Logic to the rescue

- Focus on single complex cognitive abilities.
- Requires large small amounts of training data.
- Lacks transparency/interpretability.
- Its answers can not be fully trusted.
- Not prone to data biases.
- Difficult Straightforward to incorporate background knowledge.
Via logic, we can indeed learn (deep) classifiers!

- (Deep) classifier learnability under unknown logical theories.
- (Deep) classifier learnability under probabilistic logical theories.

Via logic, we can overcome data biases!

Scene Graph Generation (AAAI 2023)

Figure: Recall of VCTree [13] on the 28 least frequent predicates: without NGP; with NGP. Benchmark: Visual Genome [9].
Knowledge Distillation into Deep Networks (ICML 2023)

Concordia

- First to support general first-order theories.
- Supports semi-/un-/supervised learning.

Video Activity Detection (ICML 2023)

\[
\text{SEQ}(B_1, B_2) \land \text{CLOSE}(B_1, B_2) \rightarrow \text{SAME}(B_1, B_2) \\
\text{DOING}(B_1, A) \land \text{SAME}(B_1, B_2) \rightarrow \text{DOING}(B_2, A)
\]

Accuracy over 5 runs

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg (%)</th>
<th>Max (%)</th>
<th>Min (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACD+L [12]</td>
<td>86.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MobileNet</td>
<td>90.07</td>
<td>91.36</td>
<td>89.61</td>
</tr>
<tr>
<td>IARG(MobileNet) [10]</td>
<td>90.18</td>
<td>92.39</td>
<td>87.55</td>
</tr>
<tr>
<td>Concordia(MobileNet, L)</td>
<td><strong>90.73</strong></td>
<td><strong>93.19</strong></td>
<td><strong>89.54</strong></td>
</tr>
<tr>
<td>Inception</td>
<td>89.72</td>
<td>91.83</td>
<td>86.84</td>
</tr>
<tr>
<td>IARG(Inception) [10]</td>
<td>88.88</td>
<td>91.67</td>
<td>85.33</td>
</tr>
<tr>
<td>Concordia(Inception, L)</td>
<td><strong>92.75</strong></td>
<td><strong>93.34</strong></td>
<td><strong>92.31</strong></td>
</tr>
</tbody>
</table>

### Entity Linking (ICML 2023)

**Table:** Results on entity linking.

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT (sp)</td>
<td>0.88</td>
<td>88.5</td>
</tr>
<tr>
<td>Concordia(BERT) (sm)</td>
<td>0.91</td>
<td>91.4</td>
</tr>
</tbody>
</table>

Leon Jonathan Feldstein, Jurčius Modestas and Efthymia Tsamoura. *Parallel neurosymbolic integration with Concordia.* In ICML, 2023 *(to appear).*
Q(O) ← NAME(herbivore, O)
NAME(N, O) ∧ NAME(N’, O) → ISA(N’, N)
→ ISA(giraffe, herbivore)
→ ISA(deer, herbivore)

Table: Recall@5 on VQAR [7].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C5</td>
<td>64.05%</td>
<td>74.62%</td>
<td><strong>87.01%</strong></td>
</tr>
<tr>
<td>C6</td>
<td>56.51%</td>
<td>72.04%</td>
<td><strong>85.45%</strong></td>
</tr>
</tbody>
</table>
Part II: Reasoning at Scale via Trigger Graphs

Trigger Graphs: Why

– Key to support goal-driven QA over transitive rules.

– Standard bottom-up evaluation:
  – may derive logically redundant facts;
  – may try to execute rules that derive no facts.

– The above negatively impact the runtime and the memory.
How: Trigger Graphs

Rules

\[ r(X, Y) \rightarrow R(X, Y) \]  \( (r_1) \)
\[ R(X, Y) \rightarrow T(Y, X, Y) \]  \( (r_2) \)
\[ T(Y, X, Y) \rightarrow R(X, Y) \]  \( (r_3) \)
\[ r(X, Y) \rightarrow \exists Z.T(Y, X, Z) \]  \( (r_4) \)

Facts

\[ \rightarrow r(c_1, c_2) \]
How: Trigger Graphs

Rules

\[
\begin{align*}
    r(X, Y) & \rightarrow R(X, Y) & (r_1) \\
    R(X, Y) & \rightarrow T(Y, X, Y) & (r_2) \\
    T(Y, X, Y) & \rightarrow R(X, Y) & (r_3) \\
    r(X, Y) & \rightarrow \exists Z. T(Y, X, Z) & (r_4)
\end{align*}
\]

Facts

\[
\rightarrow r(c_1, c_2)
\]

Bottom-Up evaluation

\[
\begin{array}{c}
    r(c_1, c_2) \\
    T(c_2, c_1, n_1) \quad R(c_1, c_2) \\
    T(c_2, c_1, c_2) \\
    R(c_1, c_2)
\end{array}
\]

November 24, 2023
How: Trigger Graphs

Rules

\[ r(X, Y) \rightarrow R(X, Y) \] (r1)
\[ R(X, Y) \rightarrow T(Y, X, Y) \] (r2)
\[ T(Y, X, Y) \rightarrow R(X, Y) \] (r3)
\[ r(X, Y) \rightarrow \exists Z. T(Y, X, Z) \] (r4)

Facts

\[ \rightarrow r(c_1, c_2) \]
How: Trigger Graphs

Rules

\[ r(X, Y) \rightarrow R(X, Y) \quad (r_1) \]
\[ R(X, Y) \rightarrow T(Y, X, Y) \quad (r_2) \]
\[ T(Y, X, Y) \rightarrow R(X, Y) \quad (r_3) \]
\[ r(X, Y) \rightarrow \exists Z.T(Y, X, Z) \quad (r_4) \]

Facts

\[ \rightarrow r(c_1, c_2) \]
Trigger graph-based reasoning

TGs delineate the rule executions

– Execute $r_1$ over the input instance.
– Execute $r_2$ over the derivations of $r_1$.
– No other operation is taking place.

Important to node

– Facts are stored inside the nodes, i.e., not stored in a single set like in all bottom-up engines.
– This data separation makes joins run faster.
Trigger graph-based reasoning

Rules

\[ r(X, Y) \rightarrow A(X) \ (r_1) \]
\[ r(X, Y) \rightarrow A(Y) \ (r_2) \]
\[ A(X) \land s(X, Z) \rightarrow T(Z) \ (r_3) \]
Trigger Graphs for Linear Rules

– **Phase I: Static TG Computation.**
  – Compute a representative instance $B^*$, i.e., one that captures all possible rule execution paths.
  – Compute a plan $G$ that mimics the rule execution when reasoning over $B^*$.

– **Phase II: Redundancy Elimination.**
  – Eliminate nodes that lead to redundant facts (via detecting preserving homomorphisms).

– **Phase III: Reasoning.**
  – The computed TG can be used to reason over all input instances.
Trigger Graphs for Linear Rules: Complexity

Let $P$ be a linear program that admits a finite universal model.

**Theorem (Complexity)**

*Computing a TG for $P$ is double exponential in $P$. If the arity of the predicates in $P$ is bounded, the computation time is (single) exponential.*
Reasoning over Linear Rules

Total materialization times in s

Pick memory in GB
Trigger Graphs for Datalog Rules

TGs for Linear Rules
- Static TG computation.
- Use the pre-computed TG to reason over all instances.
- Redundancy elimination via detecting preserving homomorphisms.

TGs for Datalog Rules
- Interleave TG creation with reasoning.
- The computed TG can be used to reason over the given instance only.
- Redundancy elimination via query containment [2].
Trigger Graphs for Datalog Rules: Example

**Rules**

\[ r(X, Y) \rightarrow S(X, Y, X) \quad (1) \]

\[ a(X) \land r(X, Y) \rightarrow S(X, X, Y) \quad (2) \]

\[ S(X, Y, Z) \rightarrow A(X) \quad (3) \]
Trigger Graphs for Datalog Rules: Example

Trigger Graph

Query for $v_3$

$Q(X) = \exists Y. r(X, Y)$

Query for $v_4$

$Q'(X) = \exists Y. a(X) \land r(X, Y)$
Trigger Graphs for Datalog Rules: Results

Let $P$ be a Datalog program.

Theorem (Soundness)

For a TG $G$ for $P$, $\text{minDatalog}(G)$ is a TG for $P$.

Theorem (Minimality)

Any TG for $P$ has at least as many nodes as $\text{minDatalog}(G)$.

Theorem (Complexity)

Deciding whether $G$ is a TG of minimum size for $P$ is co-NP-complete.
More: TG-Aware Rule Execution Strategy
Datalog Reasoning with Trigger Graphs

<table>
<thead>
<tr>
<th></th>
<th>1B</th>
<th>2B</th>
<th>4B</th>
<th>8B</th>
<th>17B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime (s)</td>
<td>203</td>
<td>226</td>
<td>520</td>
<td>993</td>
<td>2272</td>
</tr>
<tr>
<td>Memory (GB)</td>
<td>23</td>
<td>34</td>
<td>49</td>
<td>98</td>
<td>174</td>
</tr>
<tr>
<td>#IDPs</td>
<td>1B</td>
<td>2B</td>
<td>5B</td>
<td>10B</td>
<td>20B</td>
</tr>
</tbody>
</table>

**Table:** Reasoning over LUBM for 1B–17B of database triples.
Datalog Reasoning with Trigger Graphs

Materialization times in s

Pick memory in GB
Datalog Reasoning with Trigger Graphs

Materialization times in minutes

<table>
<thead>
<tr>
<th></th>
<th>VLog</th>
<th>RDFox</th>
<th>X</th>
<th>TGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claros-L</td>
<td>7</td>
<td>41</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>Claros-LE</td>
<td>46</td>
<td></td>
<td>17.5</td>
<td></td>
</tr>
</tbody>
</table>

Pick memory in GB

<table>
<thead>
<tr>
<th></th>
<th>VLog</th>
<th>RDFox</th>
<th>X</th>
<th>TGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claros-L</td>
<td>3</td>
<td>5.4</td>
<td>6.4</td>
<td>6</td>
</tr>
<tr>
<td>Claros-LE</td>
<td>11.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reasoning at Scale: How Lineage Trigger Graphs

Aim

– Develop highly-scalable reasoning techniques that support uncertainty.
– Adopt well-established semantics.
Key Challenge: Complexity

Rules

\[ e(X, Y) \rightarrow p(X, Y) \]

\[ p(X, Z) \land p(Z, Y) \rightarrow p(X, Y) \]

Facts

\[ \rightarrow e(a, b) \quad \rightarrow e(a, c) \]

\[ \rightarrow e(b, c) \quad \rightarrow e(c, b) \]

Derivations

\[ \tau_1 p(a, b) \quad \tau_2 p(b, c) \quad \tau_3 p(a, c) \quad \tau_4 p(c, b) \]

\[ \tau_5 p(a, c) \quad \tau_6 p(b, b) \quad \tau_7 p(a, b) \]
Prior Art: Key Limitations

– Relies on provenance semirings [5], i.e., associates a Boolean formula to each derivation.
– Super-polynomial size blowup in data complexity: any monotone formula to test connectivity in a graph with $n$ nodes has size $n^{\Omega(\log n)}$ (lower bound holds even for undirected graphs) [8].
– Requires Boolean checks at each reasoning step for termination.
– Runtime bottleneck.

## Probabilistic Reasoning via Provenance Semirings

<table>
<thead>
<tr>
<th>R</th>
<th>Derivation</th>
<th>Comparison</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$e(a, b)$</td>
<td>$\emptyset$</td>
<td>$e(a, b)$</td>
</tr>
<tr>
<td>2</td>
<td>$e(a, c) \land e(c, b)$</td>
<td>$e(a, c) \land e(c, b) \equiv e(a, b)$</td>
<td>$e(a, c) \land e(c, b) \lor e(a, b)$</td>
</tr>
</tbody>
</table>

### Diagram

```
\[\tau_5 \quad p(a, c) \quad \tau_6 \quad p(b, b) \quad \tau_7 \quad p(a, b)\]

\[\tau_1 \quad p(a, b) \quad \tau_2 \quad p(b, c) \quad \tau_3 \quad p(a, c) \quad \tau_4 \quad p(c, b)\]

\[e(a, b) \quad e(b, c) \quad e(a, c) \quad e(c, b)\]
```
Lineage Trigger Graphs

- Efficient maintenance of derivation history.
- Natural for TGs.
- Storing pointer offsets.
- Reduces termination checks for detecting cyclic derivations!
- No Boolean checks are required!
Lineage Trigger Graphs: (Adaptive) Provenance Circuits

– Extended the notion of provenance circuits [3] to allow a more space-efficient reasoning:
– Polynomial size representation.
Probabilistic Datalog Reasoning with Trigger Graphs

Figure: Time in seconds for goal-driven QA over sample queries from VQAR [7].
Cool Research not Covered
Goal-driven QA over existential rules with equality (AAAI 2018)

Figure: Time in msec to answer the ChaseBench queries [1].

Thanks!

Contact info: efi.tsamoura@samsung.com.


References II


Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si. 
Scallop: From probabilistic deductive databases to scalable differentiable reasoning. 

Mauricio Karchmer and Avi Wigderson. 
Monotone circuits for connectivity require super-logarithmic depth. 

References V

