Continuous Graph Pattern Matching Over Knowledge Graph Streams

Syed Gillani, Gauthier Picard, Frederique Laforest

Laboratoire Hubert Curien & Institute Mines St-Etienne, France

DEBS 2016
[Outline]

✓ Knowledge Graph (KG) Processing in general
✓ KG Streams’ Models
✓ Issues and Challengers for Processing KG Streams
✓ Pre-processing and pruning of KG events
✓ Event-based KG Stream Processing
✓ Incremental KG Stream Processing
✓ Empirical Evaluation
[The Data Deluge]

• More than 3000 Exabytes (billions GBs) created in 2015 alone
  - Increased from 150 Exabytes in 2005

• Many new sources of data become available
  ✓ Sensors, mobile devices
  ✓ Web feeds, social networks
  ✓ Surveillance video and audio
  ✓ Knowledge Bases

• **Making sense of all data: Stream Processing to the Rescue**
  ✓ Process data streams on the fly without storage
  ✓ Limited amount of available memory
  ✓ Latency of data processing matters
Stream Processing: is it enough?

- **Nature of the Streams**
  - ✓ Heterogeneous stream emanating from multiple sources
  - ✓ Extracting the contextual Knowledge
  - ✓ Seamless Integration of streams

- **Knowledge Graph Data Model**
  - ✓ Lifting streaming data to a semantic model
  - ✓ Schema-less model allows integration of heterogeneous streams
  - ✓ Integration of external sources using Link Data collections
  - ✓ New breed of applications
Knowledge Graph Model

- Set of entities and directed relations between them
- Constraints on type and attributes of the entities and their relations
- **For RDF model:**
  - Entities are IRIs, Blank Nodes, Literals (only for outer edges)
  - Relations are IRI’s
  - Set of triples *(subject, predicate, object)*
Pattern Matching/Subgraph Isomorphism (homomorphism)

✓ NP-Complete Problem
✓ Require sophisticated Indexing
Adding Temporal Dynamics to KGs

**Event/batch-based Model**

\[(\text{KG}_1, \text{T}_1)\]

\[(\text{KG}_2, \text{T}_2)\]

\[(\text{KG}_3, \text{T}_3)\]

**Edge/Incremental Model**

\[(\Delta \text{KG}_1, \text{T}_1)\]

\[(\Delta \text{KG}_2, \text{T}_2)\]

\[(\Delta \text{KG}_3, \text{T}_3)\]
[KG Stream Processing]

Query Graph

(KG₁,T₁)  (KG₂,T₂)  (KG₃,T₃)

KG Streams

Match!!
Traditional Static/dynamic Solutions

- Graph-based storage and exploration-based querying
- Tabular-based storage and join-based querying
- Re-evaluation of computed query matches

Both techniques utilised **index-store-query** model

- Expensive indexing to accelerate query processing ($O(n^4)$)
  - Clustered B+- Trees

*Require on-the-fly KG stream processing*

- Avoid expensive indexing
- Light-weight data structure
- Incremental computation of matches
What We Offer!!

✓ Continuous GPM over both Event and Incremental Model for Tumbling Windows

✓ Query-based graph pruning techniques for KG events

✓ Hybrid *join-and-explore* matching technique to avoid expensive indexing

✓ Light-weight *multi-bidirectional* data structure to comply with streaming settings

✓ Automata-based executional framework for processing of KG events
Problem 1 (Event-based CGPM). Given (i) a query graph $Q$, (ii) a Kg stream $G$, and (iii) a matching function $M$, Event-based CGPM amounts to continuously compute the function $M(Q, (G^i, \tau_i))$ for each event within the Kg streams.
Pruning KG Events

Observation 1. Given a query graph $Q$ and a KG event $(G^i, \tau_i)$, the number of edges $|E_Q| \in Q$ is less than or equal to the number of edges $|E^i| \in G^i$.

Queries are register beforehand!!

Utilise structural attributes of query graph for pruning
Pruning KG Events

$G_i$

Query-based Pruning

Materialisation into Vertically Partitioned Tables

Dictionary Encoding (Strings to Numeric Ids)
[TP-Join Automata]

✓ Automate an old friend of pattern matching
✓ Percolation property for on-the-fly processing
✓ Map the set of triple patterns (tp) to automaton states
✓ Triple patterns’ join conditions as transition predicates

Bidirectional Multimaps

\[
\begin{array}{c|c|c}
4 & S & O \\
1 & 5 \\
2 & 6 \\
3 & 7 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
13 & S & O \\
5 & 10 \\
5 & 11 \\
6 & 12 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
17 & S & O \\
10 & 15 \\
11 & 16 \\
\end{array}
\]
TP-Join Automata

✓ Hybrid Join-and-explore
✓ Join the tables for the dependent triple patterns
✓ Transit to the next state if join produces results
✓ Insert the resulted matches in graph-structures (multimaps)
✓ Explore the graph to produce the matches without creating/using indices
**[TP-Join Automata]**

- On-the-fly execution

- Each step reduces the search space by removing the "dangling" triples

- Support of start, chain, cyclic queries without incurring the cost of indexing

- Can extend the automata for expressive operators, such as kleen-+, negation

- Process joins only if there is enough evidence of matching a KG event
Incremental Continuous Graph Pattern Matching

Problem 2 (Incremental CGPM). Given (i) a query graph $Q$, (ii) an evolving KG $G$, and $(\Delta G^i, \tau_i)$ as updates to $G$, such that the updates conform to a stream $\mathcal{G} = \{(\Delta G^1, \tau_1), \ldots, (\Delta G^n, \tau_n)\}$, and (iii) a matching function $M$, Incremental CGPM amounts to continuously compute the changes $\Delta M_i = M(Q, (\Delta G^i, \tau_i))$ to the matches such that,

$$M(Q, \bigcup_{k=0}^{i-1} (\Delta G^k, \tau_k) \oplus (\Delta G^i, \tau_i)) = M(Q, \bigcup_{k=0}^{i-1} (\Delta G^k, \tau_k)) \oplus \Delta M_i$$

where operator $\oplus$ incrementally applies changes to the matched graphs.
[Incremental CGPM]

• Same approach for pruning events/graph updates

• Extend TP-Join Algorithm:
  ✓ incrementally locate new matches
  ✓ efficiently update the old matches

• Matches emerge slowly during incremental evaluation

• Find partial matches for each update and incrementally process remaining matches

• Lazy Evaluation of joins
[Incremental CGPM]

\[\text{FT} \]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{pic} \\
\hline
\text{P1} & \text{Pic1} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{n} \\
\hline
\text{P1} & \text{N1} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{frnd} \\
\hline
\text{P1} & \text{P2} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{pst} \\
\hline
\text{P1} & \text{Pst1} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{p1} \\
\hline
\text{P1} & \text{Pic1} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{n} \\
\hline
\text{P1} & \text{N1} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{frnd} \\
\hline
\text{P1} & \text{P2} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{S} & \text{O} & \text{pst} \\
\hline
\text{P1} & \text{Pst1} \\
\end{array}\]
[Incremental CGPM]

• Lazy Evaluation:
  ✓ Defer the joining process
  ✓ Make sure all the triple pattern has corresponding triples
  ✓ Store the matched results in Final Tables (FT) for a defined window

• Utilise final tables to incrementally match the new updates

• Lazy evaluation save useless computations

• Previously matched results are store in Final Tables:
  ✓ Incremental CGPM produces the same result as that of re-evaluation
[Empirical Evaluation]

✓ How the system performs as compared to traditionally Index-based solutions
✓ How the system performs as compared to re-evaluation based systems
Empirical Evaluation

• **Metrics Event-based Evaluation:**
  
  ✓ Varying the number of events and then triples within each event

  ✓ Window size would not effect the performance (no aggregate operators used)

• **Metrics Incremental Evaluation:**

  ✓ Varying the window size (w) and evaluate events within the tumbling window

  ✓ Size of the window has direct impact on the performance
Empirical Evaluation

- Datasets:
  - ✔ NY Taxi dataset with 50 million taxi related events, each event containing 24 triples
  - ✔ Social Network Benchmark (SNB) containing 50 million triples

- Queries:
  - ✔ Three NY Taxi data queries containing start, chain and combination of both
  - ✔ Three SNB queries from the use cases described in the benchmark (star, chain and cyclic)
  - ✔ Three LsBench Queries customised for SNB dataset (Used for RSP)
[Event-based Evaluation (NY-Taxi)(NY-Q1)]

![Graph showing throughput vs. number of events processed for various systems.](image)

- **E-CGPM**: Set-sized sized events
- **CQELS**: Optimisation has not much of effect

**✓** CQELS: RDF stream processing system

**✓** Jena, Sesame, RDFox: In-memory triple stores and static RDF data processing
Event-based Evaluation (SNB-Q1)

✓ Variable sized events
✓ E-CGPM shows considerable performance improvements

Table 2: Dataset distribution for large-scale CGPM, Min and Max describes the range of no. of triples for each event of SNB streams

<table>
<thead>
<tr>
<th>Dataset (Streams)</th>
<th>Min (Triples/Event)</th>
<th>Max (Triples/Event)</th>
</tr>
</thead>
<tbody>
<tr>
<td>500P</td>
<td>783</td>
<td>148K</td>
</tr>
<tr>
<td>1KP</td>
<td>2340</td>
<td>397K</td>
</tr>
<tr>
<td>5KP</td>
<td>217K</td>
<td>301K</td>
</tr>
<tr>
<td>10KP</td>
<td>50K</td>
<td>805K</td>
</tr>
<tr>
<td>20KP</td>
<td>115K</td>
<td>1.9M</td>
</tr>
<tr>
<td>30KP</td>
<td>145K</td>
<td>3.2M</td>
</tr>
</tbody>
</table>
Event-based Evaluation (SNB-Q1)

Query Processing Time

Latency (Data insertion time)
✓ Incremental Vs re-evaluation techniques

✓ Linear response with the increase in the window size
CQELS RSP system performs poorly for large datasets
Lazy Evaluation pays off with less number of join operations
[Conclusion]

- Expensive indexing-based solutions add quite a lot of latency for KG streams.
- By leveraging the hybrid join-and-explore technique such latency can be reduced.
- On-the-fly processing goes KG streams requires customised data structures.
- Incremental Evaluation outperforms re-evaluation techniques by an order of magnitude.
[Questions?]

Contact: Syed Gillani
syed.gillani@univ-st-etienne.fr