Estimating the Number of Answers to CQs using Graph Summarisation

Giorgio Stefanoni, Egor V. Kostylev, Boris Motik

Problem Statement

- **Inputs:**
  - RDF graph $G$
  - Full conjunctive query $q$ (without projection)
- **Output:** an estimate of the number of answers to $q$ on $G$
- Very important problem in practice with many applications:
  1. Show the total number of answers to the user to aid browsing
  2. Query planning!

Existing Approaches

$q = R(a, Y) \land S(Y, Z)$

1. Gather statistics about the graph; for example:
   - Count the edges of each type: $T^R = |R|$ and $T^S = |S|$
   - Count the variability of each relation’s column
     $V^R,2 = |\{Y \mid \exists X. (X,Y) \in R\}|$
     $V^S,1 = |\{Y \mid \exists Z. (Y,Z) \in S\}|$
2. Combine statistics into an estimate; for example:
   - $estimate(q) = \frac{\mu}{\max\{V^R,2, V^S,1\}}$

Cardinality estimation recognised as a critical open problem in DBs
- Estimations rely on uniformity, independence, and containment assumptions ⇒ often fail in practice
- Result can depend on the order in which we process the query
- Estimations can be way off mark, particularly for ‘long’ queries
- ⇒ No formal underpinning (just a hack!)

Main Idea and Goals

1. Use a multi-dimensional statistical representation of the graph:
   ⇒ summarise the graph by merging nodes together

![Graph $G$](image)

![Summary $S$](image)

![Mapping](image)

2. Semantics: summary represents all possible ‘compatible’ graphs

3. Estimate the expected answer over all ‘compatible’ graphs

$$E_q = \frac{q^{G_1} + q^{G_2} + q^{G_3} + \cdots}{\# \text{ of compatible graphs}}$$

4. Compute the standard deviation $\sigma_q$ and use Chebyshev’s inequality (1) to establish confidence intervals:

$$P(|X_q - E_q| \geq k \times \sigma_q) \leq \frac{1}{k^2}$$

Comparing estimation errors:

- **Linear (5)**
- **Star (7)**

Comparison Systems:

- RDF-3X: an RDF system with estimator tailored to RDF workloads
- PG-VP: PostgreSQL with vertical partitioning storage
- PG-TT: PostgreSQL with triple table partitioning storage
- **Meta**: the best of the above three systems
- **Summary**: our implementation

Computing Expectation and STD Deviation

- We give a closed-form formula to compute $E_q$
- Computing $E_q$ requires iterating only over the summary $S$, not over the ‘compatible’ graphs
- For example, $E_q = \sum_{q_1} \sum_{q_2} \frac{R^S(\alpha_1, \alpha_2) \times S^R(q_1, q_2)}{|\{\alpha_1\} \times |\{\alpha_2\}|}$
- Computing $E_q$ is intractable, but:
  - Intractable in query and summary size
  - Polynomial for CQs of bounded hypertreewidth without unification
  - Idea for handling unification: each group of unifiable atoms must be ‘covered’ by one decomposition node
- **Standard deviation** can be computed as $\sigma_q^2 = E_{q'^2} - (E_{q'})^2$, where $q'$ is the query obtained from $q$ by replacing each variable with a fresh one.

Summarising Graphs

- We summarise RDF graphs based on structural similarity:
  - Assign to each vertex $v$ a type $T(v)$ and a neighbourhood $N(v)$
  - Similarity based on Jaccard index $J(v_1, v_2) = \frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$
- **Basic algorithm:**
  - Merge $v_1$ and $v_2$ s.t. $T(v_1) = T(v_2)$ and $J(v_1, v_2)$ is largest
  - Repeat until the number of edges falls below a threshold
- Naïve algorithm is quadratic and can only handle very small graphs.
  ⇒ Efficient approximate solution using min-hashing

Preliminary Evaluation

Waterloo SPARQL Diversity Test Suite:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>WatDiv-100</th>
<th>Summary</th>
<th>Compression ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertices</td>
<td>1,053,224</td>
<td>539</td>
<td>1,954</td>
</tr>
<tr>
<td>Edges</td>
<td>10,903,668</td>
<td>9,982</td>
<td>1,092</td>
</tr>
</tbody>
</table>

![Comparing estimation errors](image)

Waterloo SPARQL Diversity Test Suite:

- **Computing Expectation and STD Deviation**
- **Summarising Graphs**
- **Preliminary Evaluation**