5-8-2007

Estimating the Cardinality of RDF Graph Patterns

Angela Maduko
Kemafor Anyanwu
Amit P. Sheth
Wright State University - Main Campus, amit.sheth@wright.edu
Paul Schliekelman

Follow this and additional works at: https://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation

This Conference Proceeding is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu, library-corescholar@wright.edu.
ABSTRACT
Most RDF query languages allow for graph structure search through a conjunction of triples which is typically processed using join operations. A key factor in optimizing joins is determining the join order which depends on the expected cardinality of intermediate results. This work proposes a pattern-based summarization framework for estimating the cardinality of RDF graph patterns. We present experiments on real world and synthetic datasets which confirm the feasibility of our approach.

2. APPROACH
In our model for RDF, nodes denote subjects or objects of schema (instance) triples while edges denote the predicates relating the respective subject and object nodes.

Figure 1: A graph pattern query and its sub-graph patterns.

2.1 Pattern Sequentialization
To summarize the frequency distribution of patterns, we desire a generic representation for all embeddings of a query pattern. To do this, we summarize the Schema graph to contain all patterns that exist and may exist in the RDF instance graph. This summary which we call the RDF Semantic and Structural Summary (GSSS) extends the RDF Semantic Summary [1]. We adopt the Minimum DFS Code (MDC) [4] of a pattern as its canonical label. Simply put, if all edges of a graph are assigned labels from a finite set of labels with a total order, the MDC of the graph is the lexicographically smallest sequence of edge labels obtained by performing a DFS traversal of the graph. (See [4] for details). We assign a 5-tuple \(<i, j, \eta(i), \eta(u), \eta(v)\>\) to each edge \(e = (u, v)\) labeled \(l\), where \(i\) and \(j\) are the DFS discovery times of \(u\) and \(v\), \(\eta\) maps \(u/v\) to the number of its GSSS node while \(\gamma\) maps \(l\) to the number of the GSSS edge of the same label.

2.2 Pattern Cardinality Estimation
Given a query pattern \(G_Q\), we first validate \(G_Q\) in GSSS to ensure it existed in the instance graph. If so, we obtain its representation \(G_{GS}\). Next, we check if the summary contains \(G_{GS}\). If so, we return its cardinality. If not, we assume that minimal size super-patterns of a pattern are uniformly distributed (the size of a pattern is the number of edges it contains). Thus, we return a product of the cardinality of its maximal size sub-pattern \(G_{GSM}\) and the pattern growth rate of \(G_{GSM}\). If \(G_{GSM}\) has D unique minimal size super-patterns (that are not in the summary) with total cardinality \(N\) and if \(G_{GSM}\) has cardinality \(C\), the growth rate of \(G_{GSM}\) given by \(N/CD\) is used to compensate for the frequency anti-monotone property which may not always hold. If \(G_{GS}\) has multiple maximal sub-patterns, we greedily select the one with the least growth rate.
2.3 Summary Construction

Our technique exploits 1) the fact that patterns may have nearly the same frequencies as their sub-patterns 2) prior knowledge of the importance of patterns. Thus the concept of a value for patterns forms the basis of our summary construction. The value of a pattern derives from its observed and estimation values. The observed value (tailored for different applications) defines known information about pattern importance. Given a set of patterns \( P = (P_1, P_2, ..., P_n) \) with frequencies \( (P_{01}, P_{02}, ..., P_{0n}) \), if \( P_{0i} = (P_{0i0}, P_{0i1}, ..., P_{0im}) \) is a vector containing the prior information about patterns such that \( 0 \leq P_{0i} \leq 1 \), we define the observed value of a pattern \( P_i (P_{OVi}) \) as \( P_{0i} \times \max(P_{Vi}) \). The estimation value of a pattern defines the number of its minimal size super-patterns whose cardinalities are estimable from its cardinality within an \( \epsilon \) error. We derive \( \epsilon \) using an exponential increase/logarithmic decrease approach to refine an initial small \( \epsilon \) until we obtain a set \( PS \) such that the total size of patterns in \( PS \) meets the budget and their total estimation value is at least the total size of patterns not in \( PS \). Based on the observed and estimation values, we value patterns as follows: Let \( P = (P_1, P_2, ..., P_n) \) be a set of patterns, let \( P_{EV\max} \) and \( P_{F\max} \) be the maximum of the expected values and frequencies of patterns in \( P \) respectively. Given a constant \( \epsilon \in [0, 1] \) and an indicator variable \( i \) that takes the value 1 if \( P_{OVi} \geq \epsilon \times P_{F\max} \) and 0 otherwise, the value of any pattern \( P_i \) is given by:

\[
P_{Vi} = (1 + PE_{Vi})(1 + PO_{Vi}) + \epsilon(1 \times P_{F\max} + 0 \times P_{EV\max})
\]

The additive constants in the first term of the equation ensure that the value of a pattern is non-zero when either its observed or estimation value is non-zero. The constant \( \epsilon \) in the second term allows the summary to be tuned to suit application needs by favoring some patterns.

Our construction algorithm first generates all patterns of up to a fixed size \( \delta \) then less valuable patterns are pruned until the budget is met. Given a budget \( B \) and vectors \( P = (P_1, P_2, ..., P_n) \), \( P_2 = (P_{02}, P_{12}, ..., P_{0im}) \) and \( P_3 = (P_{03}, P_{13}, ..., P_{0im}) \) of patterns of size at most \( \delta \), their sizes (in bytes) and their values, we consider three alternatives for selecting patterns to meet the budget.

**Optimal Solution (OPT).** The optimal solution maximizes \( \sum_0 \alpha P_{0i} \) constrained on \( \sum_0 \alpha P_{0i} \leq B \), \( (\alpha_0 \) is an indicator variable whose value is 1 if \( P_{0i} \) is part of the solution or 0 otherwise). We solve this problem using dynamic programming by observing that:

\[
V[m, B] = \begin{cases} 
V[m-1, B], & \text{if } P_{0m} > B \\
\max(V[m-1, B], V[m-1, B - P_{0m}] + P_{0m}) & \text{otherwise}
\end{cases}
\]

\( V[m, B] \) is the maximum value of patterns with total size at most \( B \). Since this solution 1) may select lower valued patterns over higher ones and 2) is NP-hard, we consider two greedy approximations.

**Maximal Value Greedy Solution (MVG).** Patterns are selected in decreasing order of value to favour less error tolerant patterns.

**Maximal Unit Value Greedy Solution (MUVG).** Patterns are selected in decreasing order of value per unit size.

3. EXPERIMENTAL EVALUATION

We experimented on real life (swetoDBLP [5]) and synthetic (Lehigh University Benchmark [2]) datasets which differ in the number and complexity of patterns. For properties (#nodes, #edges, #edge labels, average degree, #connected components (cc), min #edges in a cc, max #edges in a cc, size of patterns (bytes)), we used SwetoDBLP with (95566, 120512, 31, =3, 135, 4, 106479, 227149) and LUBM2 with (38370, 113463, 12, =6, 1, 113463, 113463, 23878). (Excluding literal triples). We compared OPT, MVG and MUVG using summary sizes of 5KB, 25KB and 50KB for SwetoDBLP and scaled down sizes of 800B, 4KB and 8KB for LUBM2 (same proportion of ratio of number of unique patterns to summary size). We also compared these with the Histogram-based solution (HS) of an RDF Database system, but for smallest summaries since we could not control the summary size of HS. We used three query workloads; 1) positive workload for patterns with cardinalities \( > 0 \); 2) frequent workload for patterns with cardinalities \( \geq 5000 \) and 3) negative workload for patterns with 0 cardinalities. Using the absolute error metric \( |C_i - C_i'| \), \( C_i \) and \( C_i' \) are the true and estimated cardinalities, we compute the percentage of patterns estimated with at most \( \epsilon \) error, \( \epsilon \in [0, E] \) where \( E \) is the maximum estimation error. We obtain pattern cardinalities for HS using the *Explain* command, after instructing the system to create detailed statistics for the datasets.

Figure 2a shows the results for the 5KB summary of SwetoDBLP, for the positive workload. HS performed worst with estimation error of at least 1024 for all patterns, compared to 20% for OPT and MVG and about 35% for MUVG. OPT, closely followed by MVG, had 100% accuracy and an error of at most 32 for 40% and 60% of the patterns. OPT performed best overall. Figure 2b shows the results for LUBM2. Due to the irregularity of LUBM2, the techniques have larger estimation errors, but given the summary sizes used, their performances are promising. For example, OPT performed best with 100% accuracy on 60% of all patterns and an estimation error of at most 1024 for 10% of patterns on the 8KB summary. We also obtained promising results for the frequent workload on both datasets. For the negative workload, HS performed best on SwetoDBLP with 100% accuracy for about 80% of patterns and worst on LUBM2, producing non-zero estimates for all patterns. OPT, MVG and MUVG produced non-zero estimates for 90% of patterns for the datasets. These false positives indicate that spurious patterns exist in G\( _{SSS} \). We note that our estimation time is negligible, 0.00031 seconds on the average.

4. FUTURE WORK

For future work, we will investigate techniques for estimating the cardinalities of larger patterns by combining those of a number of smaller ones. We also hope to improve the accuracy of negative queries and investigate how to handle updates.

5. ACKNOWLEDGMENTS

This work is funded by NSF-ITR-IDM Award #0325464 and #0714441 titled ‘SemDIS: Discovering Complex Relationships in the Semantic Web’.
6. REFERENCES


[4] Yan, X., Han, J. gSpan: Graph-Based Substructure Pattern Mining. ICDM 2002.