ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

A Survey on Approximate Queries over XML Data

Heena Malani[1], Prof. S.R Ghungrad[2]

M. E. Student, Department of Computer Science and Technology, MSS College, Jalna, Aurangabad, India[1]

Associate Professor, Department of Computer Science and Technology, MSS College, Jalna, Aurangabad, India [2]

ABSTRACT: The rapid adoption of XML as a standard for data representation and exchange prefigures a massive increase in the amount of XML data collected, maintained and queried on the Internet or in large corporate data stores. This will inevitably lead to the development of on-line decision support systems where users and analysts interactively explore large sets of XML data via a declarative query interface (eg XQuery or XSLT). Given the importance of remaining interactive, these online systems can use approximate query responses as an effective mechanism to reduce response time and provide users with early feedback. This approach has been successfully used in relational systems and is becoming more convincing in the XML world, where evaluating complex queries on structured tree data is intrinsically more expensive. Ranking and returning the most relevant query results have become the most popular paradigm in XML query processing. However, existing proposals do not sufficiently take structures into account, and therefore do not have the strength to elegantly associate structures with content to answer relaxed questions. To solve this problem, we first propose a sophisticated query relaxation framework to support approximate queries on XML data. The responses underlying this framework are not obliged to strictly satisfy the formulation of the request; Instead, they may be based on properties deductible from the original query. We thus develop a new top-k query approach that can intelligently generate the most promising responses in an order correlated to the ranking measure. We complete the work with a comprehensive set of experiments to show the effectiveness of our proposed approach in terms of precision fmeasure and recall.

KEYWORDS: Text detection, Inpainting, Morphological operations, Connected component labelling.

I. INTRODUCTION

In recent years, XML, originally proposed to represent, exchange and publish information on the Web, has become widely used in many applications: it is used as a solution for publishing inherited data, to store data that Can be represented with no other traditional data model, to ensure interoperability between different applications, to integrate Web services, to extract web data, and so on. All of these applications pose a strong demand for both repositories that store XML documents and for XML query languages, designed explicitly for XML data retrieval and restructuring. Several proposals address both problems, but all languages actually used for XML data [BC00] provide only exact answers to queries. When applied to large XML repositories or warehouses, accurate queries may require high response times. To overcome this problem, in traditional relational warehouses, fast approximate queries are supported, based on concise data statistics constructed using histograms or sampling techniques. We believe that the current trend of XML claims for extending such approaches also to the massive query of XML data sets. The basic idea for approximate answers is to store pre-calculated summaries of the XML warehouse, also called concise data collections, and query them instead of the original database, allowing Saving time and IT costs.

The basic idea for approximate answers is to store the pre- defined set of the XML warehouse, also called synopsis (concise data collections), And to query them instead of the original database, saving time and Computer costs. The user must make a request to the system, which should use the synopsis Instead of the original data to respond to the query. As result Should come in much shorter time at the cost of a loss of precision. An ideal way to respond to an XML query should benefit from the database style query and the IR style query because the IR style query improves the query value by allowing a high query level of the query. Text content To query IR style by specifying a context in which to perform the search [6].

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

The contributions of this document are as follows:

1) We propose a method of relaxation of queries integrating structures and contents, as well as the factors whose users are more concerned, for the support of approximate queries on XML data. In particular, our method assumes that users are more concerned about analyzing the original query of the user to facilitate the relaxation of the queries. Moreover, our approach differentiates the order of relaxation instead of giving equal importance to each node to be relaxed. In particular, the first relaxed structure to be considered is that which has the highest coefficient of similarity with the original query and the first node to be released is the least important node.

2) We customize the evaluation of the similarity relation by analyzing the inherent semantics presented in the XML data sources. In our solution, we classify the nodes into two groups: categorical attribute nodes and numeric attribute nodes, and design the corresponding approaches on evaluations of the similarity relationship of categorical attribute nodes and numeric attribute nodes . In addition, we design an indexed acyclic graph (CDAG) to generate and organize structural relaxations and develop an effective evaluation coefficient for the evaluation of the similarity relation on the structures.

On the basis of the evaluation of similarity proposed and the degree of importance, we complete the relaxations of the queries by an automatic recovery approach which can efficiently generate the most promising top-k responses.

II. LITERATURE SURVEY

Extensive searches were carried out on structured queries as well as text search on XML data and graph data [4]]. Structural Join [4] decomposes the tree model queries into a set of binary components, then the matches of each individual component are assembled to obtain the final results. Holistic Join [12] adopts a technique of a linked stack string to compactly represent partial results on the query paths, and these paths can be compounded to get the final matches for the tree query d 'Entrance. By introducing a fuzzy marking scheme, work [15] introduced a holistic joining algorithm to match twigs involving OR / NOT predicates. Based on the fuzzy label flows, the problem of the ordered tree pattern Concordance to fuzzy XML data has been set in the following work [12]. A common characteristic of these above approaches is that they deal exclusively with tree model queries with precise query conditions. They do not process an approximate query when users can not specify their query conditions accurately.

A system, called XRANK, designed for searching keywords in XML documents is introduced in [10]. XRANK has a ranking mechanism and returns documents as answers. Another search engine based on keywords for XML, called XSEarch, is presented in [16]. In XSEarch, query responses are classified using extended information retrieval techniques, and are generated in a similar order in the ranking. However, as mentioned earlier, querying structured data often becomes insoluble in practical applications using the above approaches, as these approaches rely heavily on schema information to avoid formulating unsatisfiable or may return unsatisfactory results. Recent efforts have combined the structured data management capabilities of databases with the powerful keyword search capabilities of information retrieval systems. In order to effectively returning the responses of the approximate structured XML query, the unstructured query mechanism [11], the keyword query mechanism [15] and the query relaxation mechanism [14] are proposed.

In [10], Brodianskiy et al. Propose a framework for deriving self-correcting queries on XML. Within their framework, satisfactory similar requests are generated when the given request is unsatisfactory. The user will then choose a satisfiable questioning of interest, and receiving exactly satisfactory answers to this request.

In [3], Mandreoli et al. Present a model that combines structures and vocabularies to support approximate queries. In [13], Liu et al. Provide a content-oriented approach to XML content and structure requests. They first decompose a query of content and structure into multiple query fragments, and get the intermediate results, and then combine and rank the results according to the relevance of the original query.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

In [11], Buche et al. Develop techniques for fuzzy marking and XML query. In [13], based on the WordNet dictionary and the fuzzy set theory [1, 2, 9], Campi et al. Extend the XPath query language to weaken query schema constraints in XML. In [18], Damiani et al. Introduce a flexible query model through a context-based aggregation choice to support approximate XML queries. In [17], based on horizon semantics, Cohen and Shiloach present a query language for XML that integrates both value-based and structural desires.

In [12] Termehchy and Winslett propose a ranking method for XML keyword search which classifies candidate responses on the basis of statistical measures of their cohesion. Their approach can allow users to leverage XQuery in processing the tree model query directly without the need for a full knowledge of checkschema.

In [18], Wang et al. Propose a method that enriches the given keywords by introducing fuzzy synonyms or keywords with semantic relations in lexical databases to improve the effectiveness of content queries only. However, simply using lexical databases fails to discover the potential similar substitutes that users may have interest in, so their approach cannot solve the problem of finding the most relevant top-k responses though.

In [15], Yan et al. Provide a preference-based ranking model to handle approximate queries in XML. Their approach relies on a user-specified interest score to generate top-k responses, which increases the burdens of user queries. It fails to discover potential responses that users may be interested in because the absence of a similarity evaluation designed to extract the inherent semantics presented in the XML data sources.

In [21], based on semi-flexible pairing and flexible matching, kanzaet al. Propose two semantics adapted to the ontological interrogation of semi-structured data. Recently, the combination of structured queries and text search to respond to approximate queries has generated much interest. In [6], Amer-Yahia et al. Introduces a framework called FleXPath that integrates structure and query into full text in XML. FleXPath interprets the XPath query as a template for keyword search, which allows for an approximate response on the structure and uses it to provide context for full-text search.

Compared to previous proposals, our approach is fundamentally different from these. In our solution, we develop a method of relaxation of queries incorporating not only the structures and the contents, but also the factors whose users are more concerned (we infer first by analyzing the original query and then by identifying the order Relaxation of structures and nodes) Respond to approximate XML queries. In addition, previous work usually gives equal importance to each node to be relaxed. However, in our method the first relaxation structure to consider is the one that has the highest tree similarity coefficient with the original query and the first node to be relaxed is the least important node. The first innovation of our solution is to introduce the factors of which the users are more concerned and to identify the order of relaxation (that is to say to relax the less important parts of the given request). Another innovation is the similarity assessment designed to discover potential responses that users may be interested in, which is customized by analyzing the inherent semantics presented in the XML data sources. In our solution, we classify the nodes in two groups: categorical attribute nodes and numeric attribute nodes. Then we design the approach on evaluations of the similarity relation of the categorical attribute nodes based on the evaluation of the percentage of ANV pairs associated with the Semantic Trees correspondents and develop the approach on the evaluation of the similarity relation of Nodes of numerical attributes based on the evaluation function.

III. DATA AND QUERY MODELING

A. Data Model We consider a data model for XML where information is represented as a series of data trees. Essentially, a data tree represents a portion of the real world through entities (usually contains a set of attributes), values and relationships among them. A simple XML data instance, which contains a heterogeneous collection of used cars, is given in Figure 1.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

XML Data Model

XML data model. Following the usual practice, we model an XML document in the form of a large T (V, E) tree, labeled by a node. Each node u ∈ V corresponds to an XML element and is characterized by a unique object identifier (oid) and a label (or tag) attributed from an alphabet of literary strings, which captures the semantics of the element. The edges (ei, ej) ∈ E are used to capture the confinement of (sub) element ej under ei in the database. (We use label (ei), children (ei) to designate the label and the set of child nodes for the element node ei ∈ V). By way of example, FIG. 1 shows an example of an XML data tree containing bibliographic data. The document consists of author elements, each with a name, and several sub-elements of paper and book. Each article contains a title, a year of publication and one or more keywords, while a book just gives its title. Note that the element nodes in the tree are named with the first letter of the element label plus a unique identifier. Leaf elements in T generally contain values, but our main objective in this work is to capture and query the tag structure of an XML data tree rather than the relevant value distributions.

XML query template. We focus on branch XML queries, which represent the basic block of declarative query languages for XML (including XQuery [4] and XSLT [7] standards). In short, a twig query describes a complex path in the XML data tree and returns a structured XML result as a tree constructed through the interleaved evaluation (that is, the structural join) of multiple path expressions (expressed in XPath [8]).

We model a query Q as a query tree labeled by node TQ, where (1) each node of TQ is labeled with a variable name qi in Q (where q0 is a distinct root node still bound to the root of the XML document);

And (2) each edge (qi, qj) of TQ is annotated with an XPath expression path (qi, qj) that describes the specific structural constraints specified in Q between the data elements related to qi and qj during 'Evaluation.

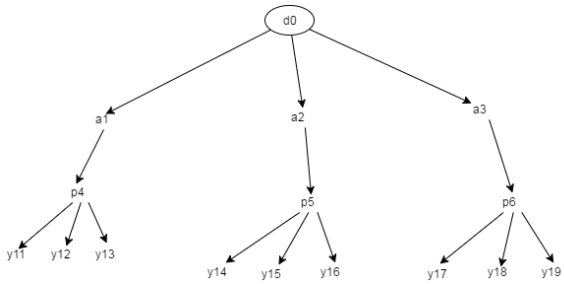

Figure XML Document

Depending on the generic notation of the trees [5], we use dotted edges to separate the paths specified in the return clause of the twig and can therefore have empty results without canceling the result of the query. We consider twig queries using XPath expressions involving only the descending and descending axes (ie, the "/" and "//" operators) and may include predicates of the form "[l¯] ", Where ¯ is usually a Path tag whose existence is required under a given parent node in the XPath expression. By way of example, the predicate "// a [// b]" in FIG. 2 specifies the author tree nodes that are located at any depth under the current link of the q0 variable (the root of the Document) and that have at least one book descendant. Intuitively, evaluating a QQ query proceeds by jointly evaluating all XPath expressions in Q on the XML tree and generating the complete set of link element tuples for Q variables. Each Binding tuple specifies Essentially an assignment of element nodes to all query variables qi so that all structural constraints specified in the query edges (qi, qj) are satisfied. We will represent the link tuples of a query Q with a nesting tree NT (Q), which contains all the elements of T that appear in the bindings of different variables and also retains their ancestor / descendant relationships as specified by the paths Of the request. Figure 2 (c) shows the nesting tree for the example request of Figure 2 (b). Obviously, the nest tree can be used to reproduce the link tuples of a query.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

Build Stable Algorithm

- *Procedure BUILDSTABLE(T)*
- *Input: XML Document T.*
- *Output: Count-Stable synopsis S of T.*
- *begin*
- *1. H := φ; S := φ*
- *2. for each element e ∈ T in post-order do*
- *3. C := {(ui, ci) : ui is a node in S and |children(e) ∩ extent(ui)| = ci > 0}*
- *4. if (H[label(e), C] = φ) then*
- *5. Add node u to S with label(u) = label(e)*
- *6. H[label(e), C] := u*
- *7. for (ui, ci) ∈ C do add edge u ci −→ ui to S*
- *8. endif*
- *9. u := H[label(e), C]; extent(u) := extent(u) ∪ {e}*
- *10. end for*
- *end*

Algorithm BUILDSTABLE

IV. PROPOSED SYSTEM

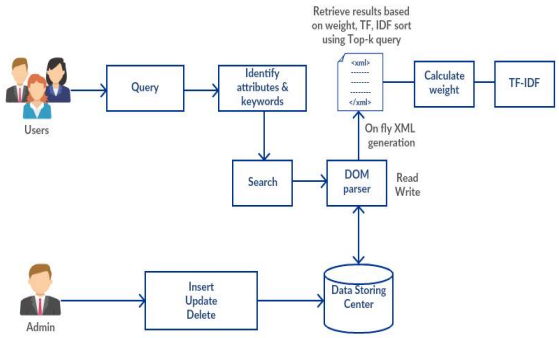

Figure 1 System Architecture

There are two users using the system

1) Data user
2) Master Admin

User query XML data. Stemming and bag of words techniques are used to identify keywords and attributes. DOM parser is used for searching attributes and keywords from XML files. To answer approximate XML queries, method calculates the weight, term frequency (TF) and inverse document frequency (IDF) of the xml file. Results are retrieved to users based on sorting of calculated weight, term frequency (TF) and inverse term frequency (IDF) based on top-k query algorithm. Information retrieval systems use various approaches to rank query answers. Users are more concerned about the most important i.e., top-k query answers in the potentially huge answer space. Different emerging applications demands for competent support for top-k queries.

Admin has writes to insert, update and delete data from data storing center. This data is used by DOM parser to retrieve results.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

Top-k query

Information retrieval systems use various approaches to rank query answers. Users are more concerned about the most important i.e., top-k query response in the potentially vast answer space. Different emerging applications demands for competent support for top-k queries. For example, in the context of the Web, the effectiveness and efficiency of meta search engines, which couples rankings from distinct search engines, are highly related to efficient rank aggregation methods. Similar applications are present in the perspective of information retrieval and data mining. Most of these applications compute queries that involve joining and aggregating multiple inputs to provide users with the top-k results.

Aim of top-k query is to retrieve best answers from a potentially large record set. Using Top-k algorithm we find the most accurate records from the given record set that matches the filtering keyword, and arrange them according to their scores. An XML tree is also generated with each result set, which is termed as a Steiner Tree. The generated Steiner tree is a list of all the records arranged according to their scores, starting from the nearest result.

Top-K query algorithm uses scores documents against keywords. Here this algorithm is used to score “Tuple Units”. A tuple unit is a set of highly relevant tuples which contain query keywords. Using these tuple units we form tuple set. A single tuple set is formed when two tuples are directly associated with each other.

Top-K query scores every tuple according to two scoring methods, namely, Direct Scoring and Indirect Scoring. Direct Scoring is based on TF-IDF algorithm. TF-IDF stands for “Term Frequency, Inverse Document Frequency”. TF-IDE presents a way to score the importance of words (or "terms") in a tuple based on how repeatedly they appear across multiple documents.

Scores are termed according to the following criteria:

1) If a word appears frequently in a tuples, it is termed as important and is scored high.

2) If a word appears in many tuples, it is termed as a unique identifier and is scored low.

Therefore, common words like "the" and "for", which appear in many tuples, will be scaled down. Words that appear frequently in a single tuple will be scaled up.

Top-K query processing algorithm works on weight-age and ranking of high frequency words and ignoring all low frequency bag-of-words. This technique is called TF-IDF (Term Frequency – Inverse document frequency). The TF-IDF weight is composed by two terms: the first computes the Term Frequency (TF), i.e. the number of times a word appears in a tuple, divided by the total number of words in that tuple row; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the tuples in the corpus divided by the number of tuples where the specific term appears.

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every tuple is different in size, it is possible that a term would appear much more times in long tuples than shorter ones. Thus, the term frequency is often divided by the tuple size.

TF(t) = (Number of times term t appears in a tuple) / (Total number of terms in the tuple).

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log(Total number of tuple-rows / Number of tuple-rows with term t in it).

Procedure

Build Tree (T)

Input: XML Document T.

Output: Count-Stable synopsis S of T.

begin

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

1. H := φ; S := φ
2. for each element e ∈ T in post-order do
3. C := {(ui, ci) : ui is a node in S and |children(e) ∩ extent(ui)| = ci > 0}
4. if (H[label(e), C] = φ) then
5. Add node u to S with label(u) = label(e)
6. H[label(e), C] := u
7. for (ui, ci) ∈ C do add edge u ci −→ ui to S
8. endif
9. u := H[label(e), C]; extent(u) := extent(u) ∪ {e}
10. end for

end

V. CONCLUSION

In this article, we presented the concepts and architecture of a system capable of performing online aggregation on XML data. This system is capable of giving rapid feedback to aggregation requests by approaching and refining the final response throughout the query processing. In addition, it provides precision guarantees by attaching weight information to estimates. We introduced a new query processing approach that divides query patterns into query layouts; An efficient query processing is performed by new operators to select and join path patterns in combination with an appropriate index structure. For precise estimates, we have adapted the principles of the DOM parser to the processing of XML queries. In the course of our in-depth evaluation, we have shown that our system refers to precise assumptions of the final response well before traditional systems can produce production. In addition, good estimates are obtained very quickly for the interrogation models without branch nodes.

REFERENCES

[1] Ashraf Aboulnaga, Alaa R. Alameldeen, and Jeffrey F. Naughton. “Estimating the Selectivity of XML Path Expressions for Internet Scale Applications”. In Proceedings of the 27th Intl. Conf. on Very Large Data Bases, 2001.
[2] P. Buneman, M. Grohe, and C. Koch. “Path Queries on Compressed XML”. In Proceedings of the 29th Intl. Conf. on Very Large Data Bases, 2003.
[3] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. “Approximate Query Processing Using Wavelets”. In Proceedings of the 26th Intl. Conf. on Very Large Data Bases, 2000.
[4] Don Chamberlin, James Clark, Daniela Florescu, Jonathan Robie, Jer´ ome Sim ˆ eon, and Mugur Stefanescu. “XQuery 1.0: An XML ´ Query Language”. W3C Working Draft, 2001.
[5] Zhimin Chen, H.V. Jagadish, Laks V.S. Laksmanan, and Stelios Paparizos. ”From Tree Patterns to Generalized Tree Patterns: On Efficient Eavluation of XQuery”. In Proceedings of the 29th Intl. Conf. on Very Large Data Bases, 2003.
[6] Zhiyuan Chen, H. V. Jagadish, Flip Korn, Nick Koudas, S. Muthukrishnan, Raymond Ng, and Divesh Srivastava. “Counting Twig Matches in a Tree”. In Proceedings of the Seventeenth Intl. Conf. on Data Engineering, 2001.
[7] James Clark. “XSL Transformations (XSLT), Version 1.0”. W3C Recommendation, November 1999.
[8] James Clark and Steve DeRose. “XML Path Language (XPath), Version 1.0”. W3C Recommendation, November 1999.
[9] Juliana Freire, Jayant R. Haritsa, Maya Ramanath, Prasan Roy, and J´er´ome Sim´eon. ”StatiX: Making XML Count”. In Proceedings of the 2002 ACM SIGMOD Intl. Conf. on Management of Data, 2002.
[10] Yannis E. Ioannidis and Viswanath Poosala. “Histogram-Based Approximation of Set-Valued Query Answers”. In Proceedings of the 25th Intl. Conf. on Very Large Data Bases, Edinburgh, Scotland, September 1999.
[11] Raghav Kaushik, Pradeep Shenoy, Phillip Bohannon, and Ehud Gudes. “Exploiting Local Similarity for Efficient Indexing of Paths in Graph Structured Data”. In Proceedings of the Eighteenth Intl. Conf. on Data Engineering, 2002.
[12] L. Lim, M. Wang, S. Padmanabhan, J.S. Vitter, and R. Parr. XPathLearner: An On-Line Self-Tuning Markov Histogram for XML Path Selectivity Estimation. In Proceedings of the 28th Intl. Conf. on Very Large Data Bases, 2002.
[13] Jason McHugh and Jennifer Widom. “Query Optimization for XML”. In Proceedings of the 25th Intl. Conf. on Very Large Data Bases, 1999.
[14] Tova Milo and Dan Suciu. “Index structures for Path Expressions”. In Proceedings of the Seventh Intl. Conf. on Database Theory (ICDT’99), Jerusalem, Israel, January 1999.
[15] N. Polyzotis and M. Garofalakis. ”Statistical Synopses for Graph Structured XML Databases”. In Proceedings of the 2002 ACM SIGMOD Intl. Conf. on Management of Data, 2002.
[16] N. Polyzotis and M. Garofalakis. ”Structure and Value Synopses for XML Data Graphs”. In Proceedings of the 28th Intl. Conf. on Very Large Data Bases, 2002.
[17] Neoklis Polyzotis, Minos Garofalakis, and Yannis Ioannidis. Approximate XML Query Answers. 2004.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor & UGC Approved Journal)

Website: www.ijirset.com

Vol. 6, Issue 8, August 2017

[18] Neoklis Polyzotis, Minos Garofalakis, and Yannis Ioannidis. “Selectivity Estimation for XML Twigs”. In Proceedings of the Twentieth Intl. Conf. on Data Engineering, 2004.
[19] C. M. Procopiuc. Geometric Techniques for Clustering: Theory and Practice. PhD thesis, Duke Univ., 2001. [20] D. Sasha and K. Zhang. Fast algorithms for the unit cost editing distance between trees. Jnl. of Algorithms, 11, 1990.
[20] Wei Wang, Haifeng Jiang, Hongjun Lu, and Jeffrey Xu Yu. Containment join size estimation: Models and methods. In Proceedings of the 2003 ACM SIGMOD Intl. Conf. on Management of Data, 2003.
[21] Yuqing Wu, Jignesh M. Patel, and H.V. Jagadish. ”Estimating Answer Sizes for XML Queries”. In Proceedings of the 8th Intl. Conf. on Extending Database Technology, 2002.