Demonstrating ProApproX 2.0: A Predictive Query Engine for Probabilistic XML

Asma Souihli Pierre Senellart
Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI
75634 Paris Cedex 13, France
first.last@telecom-paristech.fr

ABSTRACT

ProApproX 2.0 allows users to query uncertain tree-structured data in the form of probabilistic XML documents. The demonstrated version integrates a fully redesigned query engine that, first, produces a propositional formula that represents the probabilistic lineage of a given answer over the probabilistic XML document, and, second, searches for an optimal strategy to approximate the probability of the lineage. This latter part relies on a query-optimizer–like approach: exploring different evaluation plans for different parts of the formula and predicting the cost of each plan, using a cost model for the various evaluation algorithms. The demonstration presents the graphical user interface of ProApproX 2.0, that allows a user to input an XPath query and approximation parameters, and lists query results with their probabilities; the interface also gives insight into the way the computation is performed, by displaying the compilation of the query lineage as a tree annotated with evaluation operators.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases; G.3 [**Probability and Statistics**]: Probabilistic algorithms (including Monte Carlo)

General Terms

Algorithms, Design

Keywords

Approximation algorithm, cost model, probabilistic data, query processing, XML

1. CONTEXT AND CHALLENGES

Probabilistic XML suggested itself as a natural representation choice in many applications that require uncertainty management [6] and where data is already semi-structured, such as uncertain data integration, XML warehousing, uncertain version control, or Web information extraction. An example probabilistic XML document is shown in Figure 1, as an excerpt from the probabilistic *movies* database, real-life uncertain data obtained from a probabilistic data integration application [2, 3].

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.
ACM 978-1-4503-1156-4/12/10.

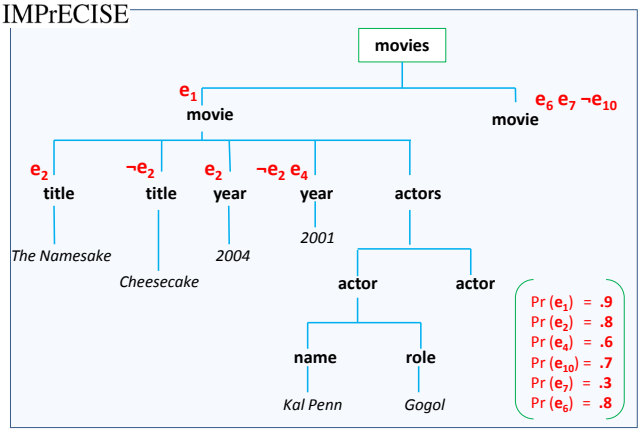

Figure 1: Abstract tree representation of an excerpt of the probabilistic *movies* database, produced by IMPrECISE [2], and translated by ProApproX into the PrXML^{cie} model

The document in Figure 1 uses a representation system with *long-distance dependencies* (PrXML^{cie} in [1, 6]), where probabilistic choices are denoted by annotating nodes of an XML tree with conjunctions of (possibly negated) Boolean random variables $e_1 \dots e_m$ called *events*; each event has a global and independent probability $\Pr(e_i)$ that e_i is true. This probabilistic XML data model, more general [1] than others used elsewhere in the literature, allows correlation of choices in different parts of the tree, by reusing event variables.

As noted in [5], there is a trade-off between strong expressive power and efficiency of query evaluation. ProApproX queries PrXML^{cie}, a more expressive model that allows dependency between nodes all over the tree, whereas other probabilistic XML systems [3, 5] consider that distant nodes in XML trees have to be independent. Our target for ProApproX was generality and wide applicability, resorting to approximation algorithms for evaluating query results, which turned out to yield competitive performance especially when it comes to scalability issues, along with a high accuracy of results.

The original data of the example in Figure 1 used a simpler, *local depeendency* probabilistic XML representation system (PrXML^{mux,ind}). ProApproX performs a tractable translation [1] to obtain the resulting tree of the figure, where uncertainty is henceforth captured by the event conjunctions.

The distributions of the *event* variables are stored elsewhere, and are gathered from the original document.

Our query language is *tree-pattern queries* extended with *joins* [6] and the major challenge behind query processing is the confidence computation of a given answer. This problem turns out to be that of evaluating the probability of a propositional formula. This is because ProApproX constructs for each query match the conjunction of all probabilistic literals on the path from the root to one of the node matched by the query. The obtained formula happens to be always in a disjunctive normal form (DNF). The problem computing its probability of a formula in DNF is known to be $\#P$-hard in general, even when all events have the same distributions [11]. This problem, however, admits a fully-polynomial randomized approximation scheme [4], as does that of querying a PrXML^{cie} document [5].

We briefly describe next the design of the approach implemented in ProApproX 2.0, for answering a query over an PrXML^{cie} document (for more detail, see the full description in [10]), before moving to the presentation of the demonstrated interface.

A companion video for this demonstration, as well as the full code of ProApproX 2.0, is freely available at `http://www.infres.enst.fr/~souihli/ProApproX2.0.html`.

2. MAIN FEATURES OF THE SYSTEM

ProApproX implements a number of exact and approximate algorithms for computing the probability of a DNF formula, each associated with a cost model [10]. This cost model was established in order to *predict* the best computation method for a given input formula. The second key principle behind the ProApproX 2.0 query engine relies on a simplification of the computation process via a decomposition of the probabilistic lineage into independent computational units [10].

Figure 2 illustrates the architecture of the system: (1) The user XPath query is translated into a different query expressed in ithe XQuery language, used to extract the probabilistic lineage. (2,3) The lineage queries are evaluated over the XML dataset through the XML::DB API for BaseX[1], the native XML DBMS. (4) The gathered probabilistic lineage is compiled by ProApproX:

Lineage preprocessing: optimizations over the DNF (removing subsumed or invalid clauses, etc.);

Compilation: repeated application of a number of operators (inconsistency, factorization, independency between clauses of the formula) to decompose the DNF into independent parts, represented as nodes of a compilation tree;

Exploration: construction of possible evaluation plans by propagation of approximation parameters down the tree under well-grounded rules and choice of (approximate or exact) evaluation algorithms at each node of the tree, the cost model being used to select the best plan found;

Computation: execution of the computation at each node and aggregation of the final result.

(5) The probability of the user query is then displayed.

We note that, with respect to the first version of ProApproX [9], in addition to adding numerous features (more complete set of evaluation algorithms, better XPath support, use of a native DBMS, support of non-Boolean queries), the optimizer of ProApproX 2.0 (cost model, lineage compilation, exploration of execution plans, evaluation on sub-formulas) is completely novel.

[1] `http://basex.org/`

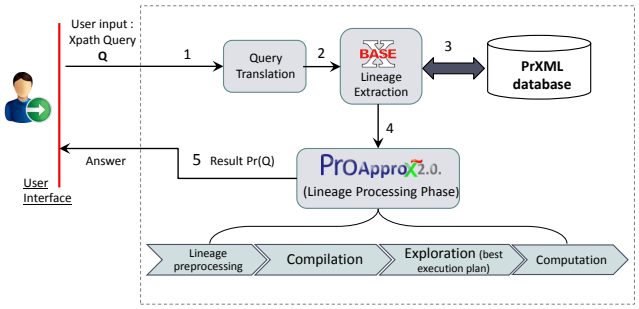

Figure 2: The architecture of ProApproX 2.0

For a glance at the evaluation performance, let us consider the most complicated query (Q_3) in the workload on the *movies* database used in [3], over the largest of the considered datasets. The DNF lineage has a total size of 47,011 literals, with 334 clauses. Run on the same desktop PC and with the same approximation precisions (multiplicative approximation, $\varepsilon = 0.1$, $\delta = 0.05$), ProApproX evaluates the probability of this query in less than half of a second, while SPROUT, the query engine of the MayBMS probabilistic relational DBMS [8], when given a query with the same lineage, went over the allocated time limit of one hour. For the same query, the running time of Trio [7], another probabilistic relational DBMS, taken directly from [3], was more than 40 seconds. We note that, for a larger workload, the performance of ProApproX 2.0 also compares favorably to that of EvalDP [5], an exact probabilistic XML query engine; indeed, this particular query cannot be evaluated using EvalDP, because of the existence of a join.

3. DEMONSTRATION

ProApproX is implemented in Java and is accompanied by a visual interface that graphically displays information about results as well as processing and computation details. The user can load a PrXML^{mux,ind} or a PrXML^{cie} database.

Figure 3 illustrates the system interface that consists of seven main panels, and an additional interface: (1) The *query editor* is used to input XPath queries on a loaded database, or on the preloaded *movies* [3] and *mondial* [5] probabilistic databases, both kindly provided by the respective authors. For example, the *movies* database features an integration of different instances of a same movie found in more than one source, stored with their respective confidences. One example query is to ask for a movie that features a given actor. (2) The *approximation settings* section allows to customize parameters for a possible approximation on the result. A combo box offers to choose between an additive or multiplicative approximation, for which an error precision ε and a confidence on the approximated result $1 - \delta$ can be customized. These parameters will only be used when the system resorts to a more efficient plan that uses approximations. (3) In the result panel, the *query information* table displays details about the total time for executing the Boolean projection of the query, and for listing different answers with their computed probabilities. The timing details report slices

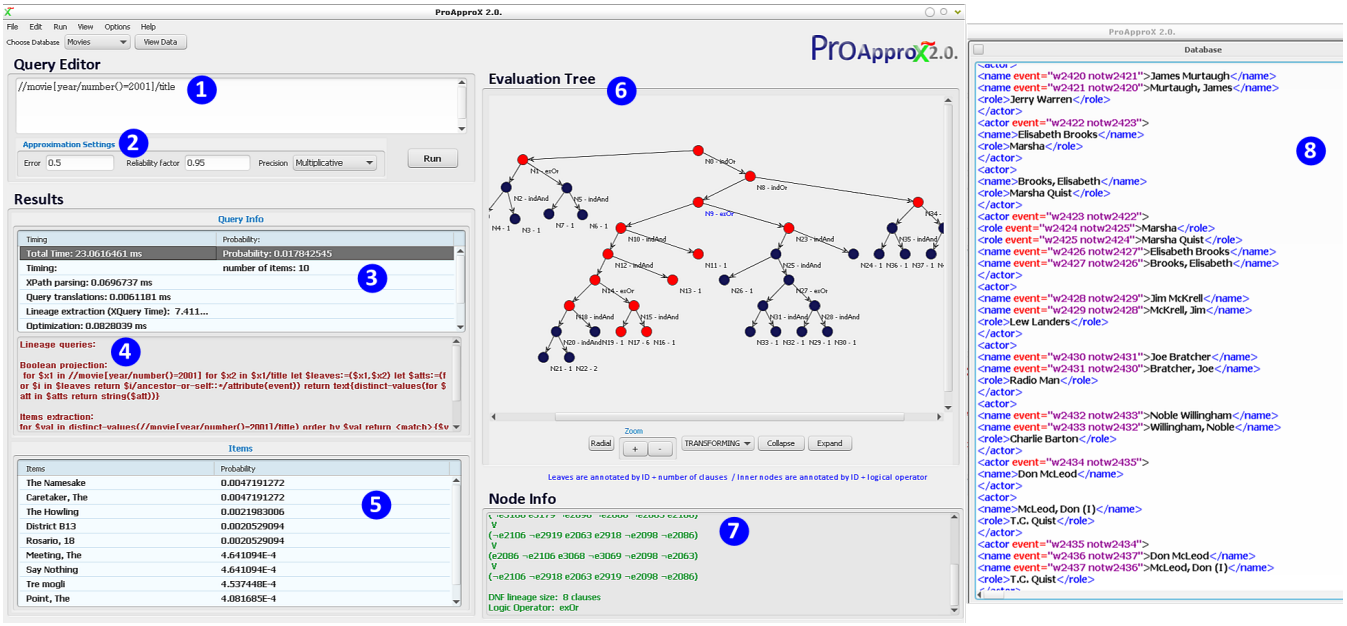

Figure 3: The user interface of ProApproX 2.0

of time spent at different steps of the evaluation process, namely: the query translation, the BaseX lineage extraction time, performance of optimizations (pre-computation phase), the compilation time, and what was spend on finding and executing the best evaluation plan. (4) The *lineage query* panel in the result section, displays the translations of the input XPath query into lineage queries. The first one is related to the Boolean projection of the query, and returns a DNF composed of a disjunction between lineages of every matching path in the tree. The second translation produces the list of result items, grouped by their values, each with its DNF lineage. (5) The *item list* table is then displayed with omputed probability values, most probable results appearing first. (6) The *evaluation tree* panel plots a graphical representation of the evaluation tree where the root node holds the DNF lineage of the query. Inner nodes store independent parts of the initial formula at that given compilation stage, with a logic operator that links the further decompositions at the children level. Ultimately, the leaves will store the last possible decomposition of a previous-level formula. The subtree shown in red is the most efficient execution plan for this query, where red nodes are assigned with computation algorithms. The evaluation of this plan gives the desired probability result. (7) By picking a node from the graphical representation, its related information will be displayed in the *node info* panel: the propositional formula and logical operator, for internal nodes, or assigned algorithm name, for leaves of the best execution plan. (8) The *database view* simply displays the currently selected XML database.

Acknowledgment

This work has been partly supported by the Dataring project of the French ANR.

4. REFERENCES

[1] S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the expressiveness of probabilistic XML models. *VLDB J.*, 18(5), 2009.

[2] A. de Keijzer and M. van Keulen. IMPrECISE: good-is-good-enough data integration. In *ICDE*, pages 1548–1551, 2008.

[3] E. Hollander and M. van Keulen. Storing and querying probabilistic XML using a probabilistic relational DBMS. In *Proc. MUD*, 2010.

[4] R. M. Karp, M. Luby, and N. Madras. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3), 1989.

[5] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query evaluation over probabilistic XML. *VLDB J.*, 18(5), 2009.

[6] B. Kimelfeld and P. Senellart. Probabilistic XML: Models and complexity. In Z. Ma, editor, *Advances in Probabilistic Databases for Uncertain Information Management*. Springer-Verlag, 2012. To appear.

[7] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, and T. Sugihara. Trio-One: Layering uncertainty and lineage on a conventional DBMS. In *Proc. CIDR*, 2007.

[8] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *Proc. ICDE*, 2010.

[9] P. Senellart and A. Souihli. ProApproX: a lightweight approximation query processor over probabilistic trees. In *Proc. SIGMOD*, 2011. Demonstration.

[10] A. Souihli and P. Senellart. Optimizing approximations of DNF query lineage in probabilistic XML, 2012. Preprint available at `http://pierre.senellart.com/publications/souihli2012optimizing.pdf`.

[11] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.