ProApproX: A Lightweight Approximation Query Processor over Probabilistic Trees

Pierre Senellart Asma Souihli
Institut Télécom; Télécom ParisTech; CNRS LTCI
46 rue Barrault, 75634 Paris, France
first.last@telecom-paristech.fr

ABSTRACT

We demonstrate a system for querying probabilistic XML documents with simple XPath queries. A user chooses between a variety of query answering techniques, both exact and approximate, and observes the running behavior, pros, and cons, of each method, in terms of efficiency, precision of the result, and data model and query language supported.

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

Probabilistic Data, XML, Query Processing, Approximations

1. INTRODUCTION AND KEY IDEA

This demonstration paper presents the implementation of a system for querying probabilistic XML data, the probabilities expressing uncertainty about the stored information. The primary goal of this work is to aid in exploratory tasks by providing quick approximate query results and statistical analysis with error bounds over discrete probabilistic XML data models. EvalDP [2] is the main existing technique for querying probabilistic XML. This polynomial-time, dynamic programming, algorithm can only be used if the data only involves *local* dependencies and if the query is a tree pattern *without value joins*. In contrast, the system demonstrated, based on the general framework of *p-documents* [1], a generalization of the different uncertain XML data models proposed in the literature, is the first able to handle *long-distance* dependencies and join queries.

P-documents are XML trees with ordinary and distributional nodes. The latter define the process of generating a random XML instance following the specified distribution at the level of each node. The model is a compact and complete representation of a probabilistic space of documents (i.e., a finite set of possible worlds, each with a particular probability). Such probabilistic space can be queried with classical tree query languages. A Boolean query over a probabilistic space, and therefore over a p-document, returns a probability, the probability that the query is true. The main idea behind the system is to efficiently calculate the probability of a given query directly on the probabilistic XML representation. This computation is exact whenever possible, or approximated using Monte Carlo methods. Previous work [2] has shown that, in general, computation is intractable under data complexity, and approximation is intractable under combined complexity. Nevertheless, several restrictions make the problem easier to tackle and many optimizations reduce the running time cost.

We emphasize that we do not deal here with the question of how a probabilistic document is obtained; this can result from a data integration process (see, e.g., [4]), from an XML warehousing application [3], from the computation of a summary of an XML corpus, etc. We assume the probabilistic document given, and we demonstrate how to efficiently query it. We explain how to compute or approximate the probability of a query using a simplified but efficient process: rewriting the initial query into one that returns every match on the underlying deterministic document as a sequence of events (labeled probabilities related to the probabilistic nodes of the p-document). These sequences are then deployed to perform exact computation or approximations.

We start by introducing an example probabilistic tree and then go into the definition of probabilistic XML and probabilistic queries. In Section 4 we explain the basics of the proposed system detailing the different evaluation methods, their advantages and their drawbacks. We conclude with the demonstration scenario. An accompanying video, showing the implemented system in action and illustrating how part of the demonstration will proceed, can be downloaded from `http://dbweb.enst.fr/proapprox.mpg`.

2. MOTIVATING APPLICATION

Uncertainty comes along with data generated by imprecise automatic tasks such as data extraction and integration, data mining, or natural language processing. A measure of this uncertainty may be induced from the trustworthiness of the resources, the quality of the data mapping procedure, etc. In many of these tasks, information is described in a semi-structured model, because representation by means of a hierarchy of nodes is natural, especially when the source (e.g., XML or HTML) is already in this form. We consider the example of a document that may result from a probabilistic data integration application such as [4].

Figure 1 shows a partial p-document, integrated from several directories providing information about person addresses.

SIGMOD'11, June 12–16, 2011, Athens, Greece.
ACM 978-1-4503-0661-4/11/06.

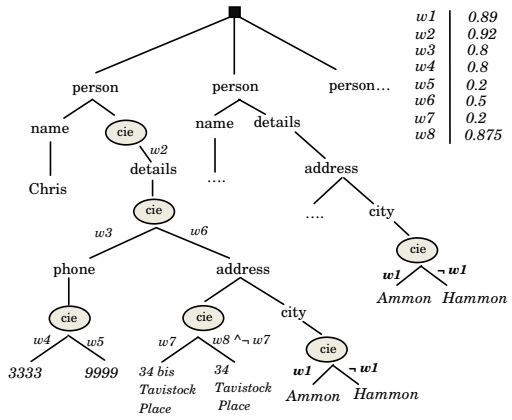

Figure 1: Portion of a probabilistic XML document resulting from probabilistic data integration.

Information about the person "Chris" may be hidden in some directories or public in some others (older or newer), which is modeled by the probability 0.92 to find details about that person. In these directories, "Chris" may be attached his first phone number (with probability 0.8), his second (with probability 0.2, independently), or both (with the probability of the independent conjunction). There are directories that assign him exclusively a first variant of his address (with probability 0.2), while others mention a second variant (with probability 0.7, mutually exclusive of the first case), or do not present any details about the address (with probability 0.1). Finally, information about Chris's city follows a general rule in the database indicating 89% of the available directories spell the name of this same city with a "H". A given directory (a possible world of the probabilistic document) only uses one of these spellings. We present in more detail in the next section the *cie* nodes and event variables used to represent such local and long-distance dependencies.

3. PROBABILISTIC DATA AND QUERIES

XML data is modeled as unranked, labeled, unordered trees. Formally, a probabilistic XML document $\tilde{\mathscr{P}}$ is a tree that consists of two types of nodes: *ordinary nodes* and *distributional nodes*. Distributional nodes are fictive nodes that specify how their children can be randomly selected. This general framework, studied in [1], is designed as a generalization of previously proposed models for PXML. Given the criteria of dependency between elements, we distinguish two types of data representation models. In the *local dependency* model, choices made for different nodes are independent or locally dependent, i.e., a distributional node chooses one or more children independently from other choices made at other levels of the tree. In the *long-distance dependency* model, which ProApproX uses to process a query, the generation at a given distributional node might also depend on different conditions (i.e., probabilistic choices) related to other parts of the *tree*. This model, more general than the local dependency model, is based on one distributional node, *cie* (for *conjunction of independent events*): nodes of this type are associated with a conjunction of independent (possibly negated) random Boolean variables $w_1 \dots w_m$ called events; each event has a global probability $p(w_i)$ that w_i is true. Note that different *cie* nodes share common events, i.e., choices can be correlated. Given a p-document $\tilde{\mathscr{P}}$, we denote by $\mathscr{P}$ the tree random variable obtained by following the random process for selecting children of each distributional node of the tree and keeping only the resulting ordinary nodes (descendant of removed nodes are also removed).

We deal with *tree-pattern* queries. For instance, given the p-document of Figure 1, we can search addresses of Chris with `//person[name='Chris']//address` in XPath notation. We also allow value joins such as in the query `//person[name=.//city]` to get all persons whose name is the same as that of their city. The semantics of such queries is standard over deterministic documents. We define the match of a query as the minimal subtree containing all nodes of a document that are mapped to a node of the query. A Boolean query over a probabilistic document returns the probability that the query is true, or, in other words, the sum of probabilities of all possible worlds in which the query is true. One fundamental observation, due to the positive, or *locally monotone* [3], nature of the query language, is that the probability of a query over a p-document $\tilde{\mathscr{P}}$ is the probability that one of the match of the query over the underlying deterministic document remains in $\mathscr{P}$. However, since the matches are not independent, the probability of a query is not the sum of the probabilities of all matches.

4. A PROBABILISTIC XML SYSTEM

Our system evaluates queries over a given probabilistic XML document either with an exact calculation when some form of independence between query matches is detected, or by running approximation algorithms elsewhere. We compare the approach to the exact EvalDP algorithm [2] (whose implementation has been kindly provided by the authors) that is very efficient but does not support either long-distance dependencies or value joins. Approximation techniques are an effective way to handle long-distance dependencies and value joins, if they can be made to run fast. This is exactly the purpose of our system, dealing effectively with local or long-distance dependencies and tree-pattern queries with joins. We have the possibility of taking as input a p-document in the *local dependency* model, with *ind* – standing for *independent choice* of each child – and *mux* nodes – standing for *mutually exclusive choice* of at most one child. As noted in [1], *ind* and *mux* can be tractably translated into *cie* nodes. We appeal to these theoretical studies to perform efficient translations, as encoding matches with conjunctions of all events involved simplify the computation and approximation processes.

Encoding the Matches. Consider now the following query: `//person[name='Chris']//address/text()` over the probabilistic document of Figure 1. We can actually discard the *cie* nodes and integrate each event w_i as an attribute attached to its corresponding XML node. We rewrite a query to form another query that retrieves the concatenation of all events appearing along a match, e.g., the address query can be rewritten to return all possible matches: $\langle w_2, w_6, w_7 \rangle$, $\langle w_2, w_6, w_8, \neg w_7 \rangle$. We run approximations directly over these event conjunctions. To evaluate the rewritten query and retrieve the different matches, the system features efficient, index-based, XQuery processing using Saxon as a Java class library.

Additive Approximation. The simplest way to implement an additive Monte Carlo approximation technique, as mentioned in [2], is to generate random instances from the p-document, evaluate the query over them, and get the approximated probability as the proportion of samples that make the query true. We implemented this technique but performance was very low due to the cost of generating a sample of a potentially very large database, and the resulting I/O operations. *Additive approximation* is reconsidered by drawing random values for the w_i's belonging to the path of each mapping, in this way getting rid of the instance generation. This gives much more acceptable running time. However, because of the very nature of additive approximation, the convergence is slow for low values of probabilities.

Multiplicative Approximation. The probability of query Q being true in a random instance $\mathscr{P}$ can be computed [2] by: $\sum_{i=1}^{n} \Pr(m_i \lhd \mathscr{P}) \times \Pr\left(\bigwedge_{j=1}^{i-1} \neg(m_j \lhd \mathscr{P}) \mid m_i \lhd \mathscr{P}\right)$, where $\Pr(m_i \lhd \mathscr{P})$ is the probability that a match m_i of Q remains in $\mathscr{P}$. The first term is easy to compute by just gathering the probabilities of all events involved in the match; the second term can be approximated by conducting biased draws to considerate the probability that none of the preceding matches exists in the random document $\mathscr{P}$, given that a current match m_i appears in that same document. The evaluation leads to very good accuracy. Nevertheless, the convergence guarantee [2] that is obtained from Hoeffding's inequality requires a running time growing in $O(n^3 \ln n)$ in the number of matches to the query (which is, itself, potentially exponential in the size of the query). However, the cases with high number of matches are, empirically, mostly those for which the probability of the query is high, and consequently when additive approximation gives good results. Therefore, our system yields processing of queries with high number of matches to the additive approximation algorithm and with low number of matches to the multiplicative approximation.

Independent Computation. Of course, a query that only involves independent matches does not require a complex process; this case often appears in practice. It suffices to note the independence between patterns to the query and to compute the result based on the principle of inclusion-exclusion. Exploring the set S of patterns to the query, we are able to detect *independence up to intersection* by simply verifying the following property (given that n is the size of S, and m_k is a match in S): *patterns to the query are independent up to intersection if* $\bigcap_{i=1}^{n} m_i = m_u \cap m_v$ *for all* $1 \leq u < v \leq n$. It suffices then to compute the global result based on the principle of inclusion-exclusion, e.g., for two independent mappings m_1 and m_2 to a given query Q and a random document $\mathscr{P}$: $\Pr(\mathscr{P} \models Q) = 1 - (1 - Pr(m_1)) \times (1 - Pr(m_2))$.

Optimizations. Several optimizations are implemented, applied mainly to *multiplicative approximation*. A first preprocessing removes mappings that are always true or false, the second goes along with the main treatment and decides to stop the computation and returns the result when the contribution of remaining mappings is considered to be very low (given a sorted input of mappings according to their probability in the tree).

We implemented an XPath parser that generates the query yielding the sequence of events for every matching pattern. In order to compare the performance of our system with EvalDP, we use the same dataset and queries as in [2]. First results show that we are able to perform faster evaluation with acceptable precision guarantees, for most of these queries. We also noticed that for some query, the implementation of the EvalDP algorithm leads to a wrong value, which may be attributed to the relatively high intricacy of implementation of this dynamic programming algorithm. We stress that our system allows us to process a wider range of queries (especially, join queries), over a wider range of models (long-distance dependencies).

5. DEMONSTRATION SCENARIO

We describe here the structure of the user interface for interacting with the system, as we intend to demonstrate it.

Input. The user can select a p-document from a default collection (including the dataset from [2] and an extension of the example of Figure 1), or upload an external one. She can specify an XPath query or choose to run a preset query from a proposed list related to the current internal p-document. Regarding the evaluation method, settings can be custom-made: the user can choose to run one or more algorithms (naïve evaluation, EvalDP, independent exact computation, approximations), or run the default configuration that picks an adequate method for the given query as explained in Section 4. We also give the possibility to personalize the number of samples needed for additive and multiplicative approximations, in one of the following three ways: (i) by calculating, using Hoeffding's inequality, a suitable number of trials given a tolerated error under a probabilistic guarantee; (ii) by empirically stopping the sampling once the estimated probability has not deviated more than a given value over a given number of consecutive trials; (iii) by giving a fixed number of trials.

Output. Different results are displayed, namely the probability of the query for each method run, running times, and probabilistic guarantees given by Hoeffding's inequality for approximation techniques. Additionally, the user can choose to plot the evolution of the estimated probability of approximation techniques and corresponding error intervals, to get a better grasp on the precision obtained by these techniques. When a technique is not applicable to a given p-document or query, the user is also informed of this fact.

6. REFERENCES

[1] S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the expressiveness of probabilistic XML models. *VLDB J.*, 18(5):1041–1064, 2009.

[2] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query evaluation over probabilistic XML. *VLDB J.*, 18(5):1117–1140, 2009.

[3] P. Senellart and S. Abiteboul. On the complexity of managing probabilistic XML data. In *Proc. PODS*, Beijing, China, June 2007.

[4] M. van Keulen and A. de Keijzer. Qualitative effects of knowledge rules and user feedback in probabilistic data integration. *VLDB J.*, 18(5):1191–1217, 2009.