A Probabilistic XML Merging Tool*

Talel Abdessalem
Institut Télécom
Télécom ParisTech; CNRS LTCI
Paris, France
talel.abdessalem@telecom-paristech.fr

M. Lamine Ba
Université Cheikh Anta DIOP
DMI; FST
Dakar, Sénégal
mouhamadou84.ba@ucad.edu.sn

Pierre Senellart
Institut Télécom
Télécom ParisTech; CNRS LTCI
Paris, France
pierre.senellart@telecom-paristech.fr

ABSTRACT

This demonstration paper presents a probabilistic XML data merging tool, that represents the outcome of semi-structured document integration as a probabilistic tree. The system is fully automated and integrates methods to evaluate the uncertainty (modeled as probability values) of the result of the merge. It is based on the two-way tree-merge technique and an uncertain data model defined using probabilistic event variables. The resulting probabilistic repository can be queried using a subset of the XPath query language. The demonstration application is based on revisions of the Wikipedia encyclopedia: a Wikipedia article is no longer considered as the latest valid revision but as the merge of all possible revisions, some of which are uncertain.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications

General Terms

Algorithms

Keywords

Probabilistic XML, XML merge, tree merge

1. INTRODUCTION

Data management in collaborative online platforms (*wikis*) is limited to the use of revision control to manage changes on documents. The revision control system maintains the current revision of each document and generates a new revision after every editing operation. This simple system works well but requires user intervention when conflicts or uncertain data are detected and does not include any mechanisms for controlling information reliability.

*This work has been supported by the DataRing and LPOD projects of the French ANR.

EDBT 2011, March 22–24, 2011, Uppsala, Sweden.

Efficient evaluation of the uncertainty in merged data and automatic resolution of conflicts are the challenging points for building a fully automated data integration system. Evaluation of uncertainty is needed in collaborative web platforms because a document can result from contributions of different persons, who may have different levels of reliability. This reliability can be estimated in various ways, such as an indicator of the overall reputation of an author (possibly automatically derived from the content of contributions, cf. [2]) or the subjective trust a given reader has in the contributor. For popular collaborative platforms, an automatic management of conflicts is necessary because the number of contributors can be very large. This is especially true for documents related to hot topics, where the number of conflicts and vandalism acts can evolve rapidly and compromise the integrity of the documents.

In the case of the online encyclopedia Wikipedia, very popular and with no write access limitations for most documents, a system able to estimate the level of reliability of information and to automatically manage conflicts would be very helpful. In a sense, a Wikipedia page may be regarded as a representation of all possible worlds modeled by the integration of generated revisions. A Wikipedia article can thus be seen as the merge of all its (uncertain) revisions. The overall uncertainty on a given part of the article is derived from the uncertainty of revisions having affected it.

In this demonstration paper, we describe a probabilistic semi-structured data merging tool, which is fully automated and provides facilities for automatically resolving conflicts and assessing uncertainty about the merged data. The system is based on a two-way probabilistic XML merging algorithm that computes a probabilistic XML document from the successive input revisions of a given document. The result document represents several possible views of the real world and enables estimating the reliability of the information it contains. Moreover, our system enables viewing the state of a document at a given revision, removing the effect of a given revision or a given contributor, or focusing only on the effect of some chosen revisions or some reliable contributors. We use a specific model for probabilistic XML, from [1, 4], based on the use of probabilistic event variables. A probabilistic document (p-document) is defined as a tree with ordinary nodes and distributional nodes, the latter specifying a probability distribution of children of a given node.

We start by presenting the probabilistic XML model of p-documents. Then, in Section 3, we describe our merging algorithm, and in Section 4 we describe the implemented system and the demonstration scenario.

2. PROBABILISTIC XML

A probabilistic XML document is a representation of a probability distribution over a space of ordinary XML documents. We follow the framework of a general probabilistic XML model, from [1], in which this distribution is modeled in terms of a probabilistic process that generates an ordinary random XML document (seen here as an unranked, labeled, and ordered trees). Note that [1] assumes unordered trees, but the same theory can be extended to ordered trees (as in [3]), that are better adapted to our application to Wikipedia revisions.

Probabilistic Documents. A p-document is a tree with two types of nodes. We have the ordinary nodes, the regular XML nodes which may appear in random documents, and the distributional nodes. The latter are only used for defining the probabilistic process that generates random documents (but they do not occur in those documents) by specifying how the children of a given node are randomly selected. Several families of probabilistic documents, characterized by different types of distributional nodes, have been proposed in the literature [1]. A general distinction is made between local distributional nodes (where the probabilistic choice is made at each node independently of all other choices) and global ones. The former tend to be more tractable query-wise, whereas the latter express more compactly the dependencies resulting from update operations [4]. Since the tree merging operations we use for obtaining a probabilistic document can be seen as a succession of updates, we focus here on global distributional nodes, and, more specifically, on the very general *fie* nodes [4] (standing for *formula of independent events*) where each distributional node is annotated with an arbitrary propositional formula over Boolean random variables. All update operations can be made in polynomial-time data complexity, with the downside that all non-trivial queries are **#P**-hard to evaluate in this model. This probabilistic XML representation system is denoted PrXML^{fie}.

Formally, a p-document $\widehat{\mathcal{P}} \in \mathsf{PrXML}^{fie}$ is an unranked, labeled, ordered, tree with ordinary and distributional nodes (the root and the leaves are assumed to be ordinary), along with a set of *probabilistic events* $\{e_1 \dots e_n\}$ (Boolean random variables), with probabilities $\Pr(e_i) \dots \Pr(e_n)$ of being true. The assumption is that these events are all probabilistically independent, i.e., $\Pr(e_i \wedge e_j) = \Pr(e_i) \times \Pr(e_j)$ for all $i \neq j$. Each distributional node n of $\widehat{\mathcal{P}}$ is associated with a propositional formula $\phi(n)$ over $e_1 \dots e_n$. The probabilistic process that generates a random document $\mathcal{P}$ from a $\widehat{\mathcal{P}}$ is as follows: (i) choose a valuation for all event variables, setting e_i to true with probability $\Pr(e_i)$ for each $1 \leq i \leq n$; (ii) remove all distributional nodes n whose formula $\phi(n)$ evaluates to false for this valuation; descendants of removed nodes are removed as well; (iii) remove all remaining distributional nodes (whose formula was evaluated to true) and connect their ordinary children to their closest ordinary ancestors.

An example PrXML^{fie} p-document, along with its three possible documents and their probabilities, is depicted in Figure 1. For concision of representation, *fie* distributional nodes are simply represented as annotations on the edges connecting their two closest ordinary ancestor and descendant. Following this convention, we will refer in the following to formulas attached to a node for the formula of a virtual distributional node in between the node and its father.

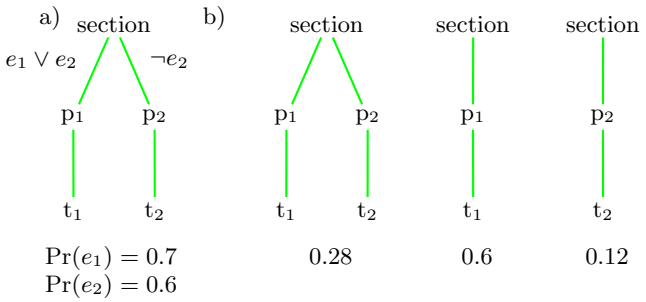

Figure 1: a) PrXML^{fie} p-document; b) its three corresponding possible documents and their probabilities

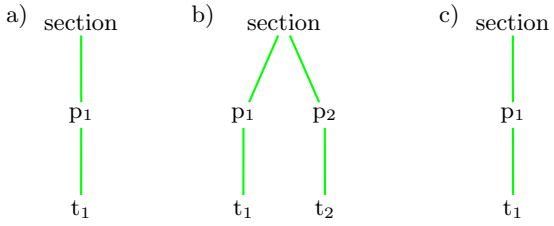

Figure 2: a) base revision; b) node p_2 is added ; c) node p_2 is then removed

P-Documents and Data Integration. A probabilistic document can be considered as the result from a number of data integration operations, as in [8, 4]. P-documents provide a concise representation of imprecision and give the possibility to keep lineage (or provenance) information about the data: probabilistic events can represent the source of each operation, and the probability in these events the confidence one has that the source is certain.

In our context of revisions in collaborative platforms, several natural semantics can be associated to the event variables. One possibility is to see each successive revision as an independent event and represent the uncertainty in this revision as the probability of this independent event. More realistically, one can use probabilistic events e_i to denote the author of a given revision and their probability $\Pr(e_i)$ the trust or reputation attached to this particular contributor. Finally, it is possible to combine the two approaches and assign to each revision a more complex event formula depending on multiple events identifying different components of the lineage of the formula.

In the management of Wikipedia revisions, a new revision of a document is generated from the current revision when a set of operations of updates is executed. Figure 2 shows a simplified example of an update operation on a Wikipedia revision where a single node is added and then removed. To integrate these revisions, which is the purpose of this work, we encode these update operations with event formulas of a global p-document, which represents the entire set of revisions.

3. MERGING PROBABILISTIC XML DATA

Our system realizes the integration of probabilistic XML documents based on probabilistic events with a two-way tree

merging technique. The merge of two successive revisions is done in two steps. Firstly, the merge algorithm finds shared, added, and deleted nodes in two successive revisions, in the same spirit as [6, 7]. Secondly, this information is merged into the global p-document that is incrementally built out of all revisions. We detail these two steps further.

For our application case, the merge of Wikipedia revisions, we implemented a search interface that allows retrieving a list of revisions of a given page from either a sample in a local database or from the online Wikipedia platform on demand. The Wiki-formatted revisions are then converted to XML by following the structure (sections, subsections, paragraphs, etc.) of the article. In Section 4, we give more details on the functionalities included in the system.

We consider the following example of a Wikipedia article with four successive revisions:

1. An introduction paragraph is added to the initially empty document.
2. A section is added after the introduction, containing some initial text.
3. This section is removed.
4. The section added in revision 2 and deleted in revision 3 is restored.

This sequence of revisions is typical, for instance, of a vandalism act that is subsequently corrected (but the vandal may either be the one that adds an irrelevant section and then restores it back after deletion, or the one that removes a valid section).

Let $r_1 \dots r_n$ be the n successive revisions of a given article (i.e., n ordinary XML documents). By convention, we assume that r_0 is a root-only document. We build iteratively, for each $1 \leq k \leq n$, a p-document $\widehat{\mathcal{P}}$ corresponding to the merge of the first k revisions. In addition to building $\widehat{\mathcal{P}}$, our algorithm maintains a function α that serves to remember the mapping between nodes of revisions x and nodes of the p-document $\alpha(x)$. Initially, $\widehat{\mathcal{P}} = r_0$ and α maps the root of r_0 to that of $\widehat{\mathcal{P}}$. We iteratively match r_{k-1} with r_k and report the result of the match into $\widehat{\mathcal{P}}$.

Matching of Revisions. Consider two two revisions r_{k-1} and r_k to be matched. Let f_k be the event formula associated to revision k (f_k describes the provenance of this revision, we explain in Section 4 how we choose it in our application). The semantics of that formula is that it evaluates to true if the revision is considered to be valid.

The matching of XML trees consists in determining a node correspondence between the two trees. We assume that only nodes at the same depth level in the trees match (i.e., for any node $n \in r_{k-1}$ and $m \in r_k$ that match, the children of n may only match children of m and vice versa). The matching algorithm gives as result three sets:

- A set D of deleted nodes: if $x \in A$ then $x \in r_{k-1}$ and x has no match in r_k.
- A set A of added nodes: if $x \in A$ then $x \in r_k$ and x has no match in r_{k-1}.
- A set M of matched couples: if $(x, y) \in M$ then $x \in r_{k-1}$ and $y \in r_k$ match.

The algorithm finds the shared nodes in the two trees by matching text content of leaf nodes, and children element sequences of internal nodes; more details can be found in [6]. Then, the deleted nodes in the first document and the added ones in the second one are discovered separately as those with no match.

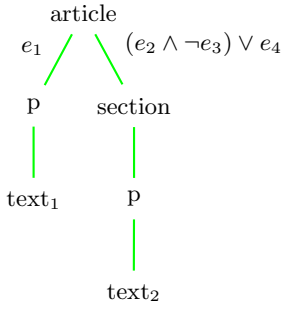

Figure 3: Example merge result p-document resulting from the merging process

Merging Matches. In this step, we update the p-document with nodes and event formulas describing the match between the two successive revisions. Let $\mathit{fie}_{\text{old}}(x)$ and $\mathit{fie}_{\text{new}}(x)$ be the event formula attached to a node $x \in \widehat{\mathcal{P}}$ respectively before and after this merge.

We proceed as follows:

- For deleted nodes $x \in S$, we set
$$\mathit{fie}_{\text{new}}(\alpha(x)) = \mathit{fie}_{\text{old}}(\alpha(x)) \wedge (\neg f_k).$$
- For $(x_{k-1}, x_k) \in M$, we set $\alpha(x_k) = \alpha(x_{k-1})$ and
$$\mathit{fie}_{\text{new}}(\alpha(x_k)) = \mathit{fie}_{\text{old}}(\alpha(x_k)).$$
- For added nodes $x \in A$, we look for the lowest ancestor a_k of x_k such that $(a_{k-1}, a_k) \in M$ for some a_{k-1}. We compare the subtree T_k rooted at a child of a_k that contains x_k with each of the subtrees rooted at children of $\alpha(a_{k-1})$. There are two possible situations:
  - We do not find a match: then we add a new node x to $\widehat{\mathcal{P}}$, with $\mathit{fie}_{\text{new}}(x) = f_k$.
  - We find a match c_{k-1}: then we set
$$\mathit{fie}_{\text{new}}(\alpha(c_{k-1})) = \mathit{fie}_{\text{old}}(\alpha(c_{k-1})) \vee f_k.$$
  In both cases, we set $\alpha(T_k)$ correspondingly.

Obvious simplifications of these formulas are made, so that, for instance, $\mathsf{true} \vee \phi$ is replaced with ϕ, or $e \vee \neg e$ is replaced with true. We also not repeat on a node the conditions of its ancestors.

The output of the whole process on the four-revision article evoked earlier on is depicted in Figure 3, when f_k is simply an independent event variable e_k for all k.

4. DESCRIPTION OF THE SYSTEM

Semantics and Use Case. In our demonstration application (Wikipedia), we associate each revision r_i of a given document with a unique event e_i, as well as with an event e'_j identifying its author. The event formula coming with the revision is $f_i = e_i \wedge e'_j$. $\Pr(e'_j)$ can be interpreted as the value of trust or reputation of the contributor, and $\Pr(e_i)$ as the probability this particular revision is correct conditioned by the fact that the author is reliable. The distribution of the $\Pr(e_i)$'s and the $\Pr(e'_j)$'s is user-defined in our application, or determined using a preset probability distribution.

The probabilistic document corresponding to a Wikipedia article can be used for a number of use cases, for instance:

- by setting the events $e_1 \dots e_i$ to true for revisions r_1 to r_i and all other e_k's to false, as well as all e'_j's to true, one can retrieve the sole content of the r_i revision;
- by setting e_i to false, one can obtain the state of the article as if the revision r_i never happened;
- by setting e'_j to false, one can remove all the revisions of the contributor associated with e'_j.

Besides these basic use cases, our technique can also be used to automatically deduct the parts of revisions or contributions whose corresponding probability value is above a fixed minimum confidence value.

More generally, given a distribution of probability on the e_i's and e'_j's, our application allows estimating the probability of parts of the article. These parts are selected with a subset of the XPath query language. For example, the execution of the XPath query //p on the p-document of Figure 3 gives as result the introductory paragraph, with probability $P(e_1)$ and the second paragraph, with probability $P((e_2 \wedge \neg e_3) \vee e_4)$. Additionally, the result of a given query can be refined according to a probability value threshold, e.g, when we want to find all parts having probability greater than 0.9. Though the problem of computing the probability of a query is in general #**P**-hard in PrXML^{fie}, the limited amount of correlations in merged documents makes the probability computation easier in practice, especially for single-path queries [4]. As a last resort, it is possible to use exhaustive enumeration of possible worlds to compute probability values. We refer to [5] for more details about query evaluation on probabilistic XML.

Implementation. Our system is implemented in the Java programming language and with the Jython API that allows to call Python functions from a Java program. The Python scripts are especially dedicated to text processing and XML document manipulation with the PyXML module. In the proposed tool the format of Wikipedia revisions is converted to an XML structure according to the format of sections and paragraphs used in this platform. As an example, a section always begins with a sequence of the special character "=" and all its successive paragraphs are separated by a blank line. An overall view of the different modules composing our system is sketched in Figure 4.

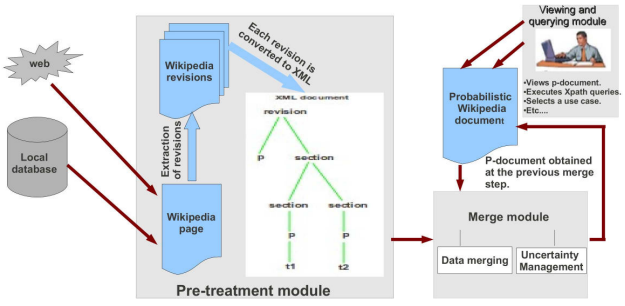

Figure 4: Architecture of the system

Demonstration Scenario. The user chooses a Wikipedia page either among those stored in a local database, or from the live Web site of Wikipedia, using keyword search. In the latter case, the revisions of the page are downloaded and stored locally for further processing. After choosing a page, the user can select all or part of its revisions (see the lower part of the screenshot). Some filtering options enable the user to select the revisions according to the contributor who is responsible for them or according to their dates. The contributors are listed with their preset trust (or reputation) values. These values can be modified online by the user. Then, the merge can be processed on the selected revisions and the name of the obtained p-document appears on the screen (see the upper-right part of the screenshot). After this, the user can select a p-document and display its content. He or she can manipulate two display options: a tree view or a text view. In both cases, when a document is displayed the probabilities associated to each part of it are indicated. The use cases listed previously (displaying a specific revision, removing it, merging different combinations of revisions, etc.) can be tested. Finally, the user can query the p-document using a fragment of the XPath query language and view the obtained result and its corresponding probability.

5. CONCLUSION

We have presented a tool for representing the merge of various revisions of a given XML document as a probabilistic XML document, which can then be queried to retrive the probabilities of some specific parts of the document. This is one of the first actual applications of the existing literature on probabilistic XML [1, 5]. The system also allows hypothetical reasoning, to see the state of an article as it were if some given revisions did not happen. Combined with a way of estimating the reputation [2] or trust in the contributors of the Wikipedia, it can be a useful and novel way to visualize an article of the online encyclopedia.

6. REFERENCES

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

[2] B. T. Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. In *Proc. WWW*, Banff, Canada, May 2007.

[3] M. Benedikt, E. Kharlamov, D. Olteanu, and P. Senellart. Probabilistic XML via Markov chains. *Proceedings of the VLDB Endowment*, 3(1), Aug. 2010. Presented at the VLDB 2010 conference, Singapore.

[4] E. Kharlamov, W. Nutt, and P. Senellart. Updating probabilistic XML. In *Proc. Updates in XML*, Lausanne, Switzerland, Mar. 2010.

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

[6] R. La Fontaine. Merging XML files: A new approach providing intelligent merge of XML data sets. In *Proc. XML Europe*, Barcelona, Spain, May 2002.

[7] T. Lindholm. A three-way merge for XML documents. In *Proc. DocEng*, Milwaukee, WI, USA, Oct. 2004.

[8] M. van Keulen, A. de Keijzer, and W. Alink. A probabilistic XML approach to data integration. In *Proc. ICDE*, Tokyo, Japan, Apr. 2005.