User Feedback in Probabilistic XML

Ander de Keijzer\textsuperscript{o} \hspace{1cm} Maurice van Keulen\textsuperscript{o}
\hspace{1cm} a.dekeijzer@utwente.nl \hspace{1cm} m.vankeulen@utwente.nl
\textsuperscript{o} Department of EEMCS
University of Twente
P.O.Box 217
7500 AE Enschede
The Netherlands

Abstract

Data integration is a challenging problem in many application areas. Approaches mostly attempt to resolve semantic uncertainty and conflicts between information sources as part of the data integration process. In some application areas, this is impractical or even prohibitive, for example, in an ambient environment where devices on an ad hoc basis have to exchange information autonomously. We have proposed a probabilistic XML approach that allows data integration without user involvement by storing semantic uncertainty and conflicts in the integrated XML data. As a consequence, the integrated information source represents all possible appearances of objects in the real world, the so-called possible worlds.

In this paper, we show how user feedback on query results can resolve semantic uncertainty and conflicts in the integrated data. Hence, user involvement is effectively postponed to query time, when a user is already interacting actively with the system. The technique relates positive and negative statements on query answers to the possible worlds of the information source thereby either reinforcing, penalizing, or eliminating possible worlds. We show that after repeated user feedback, an integrated information source better resembles the real world and may converge towards a non-probabilistic information source.

1 Introduction

A major source of semantic uncertainty and conflicts during data integration, hence for the need for human involvement, is data overlap. Data overlap occurs when data sources contain data about the same real world objects. Human knowledge is required to decide if two data items refer to the same real world object and, if so, how to resolve conflicts when merging these data items. The idea behind probabilistic data integration is to avoid human involvement during data integration by postponing the resolution of semantic uncertainties and conflicts to a moment more natural to human involvement namely during querying. This paper presents a technique that allows user feedback on query results to be used to resolve semantic uncertainties and conflicts encountered during data integration. Since we focus on data overlap, we assume that the schemas of the data sources are already aligned.

In [KKA05], we introduced a probabilistic XML model which allows storage of semantic uncertainties occurring during data integration as a means to effectively postpone semantic decisions. Probabilistic XML is an extension of XML with a few special nodes that allow to simultaneously represent possible representations of a real world object with a probability indicating the level of confidence of those possibilities. We chose XML, because the distinction between different types of uncertainty found in a relational model [ZP97], effectively disappears with XML [KKA05]. Moreover, XML is often the chosen data format for data exchange and integration anyway.

A challenge in the quest towards a DBMS self-managing data integration in this way is that without any world knowledge, almost anything, however unlikely, is theoretically possible. The challenge is to keep the integrated data to a manageable size. In [KKL06], we showed that adding just very little world knowledge in the form of generic knowledge rules can reduce the size of the integrated data enormously. In other words, probabilistic data integration in theory does not require human involvement, in practice some is needed, but only very limited human involvement suffices.

We illustrate probabilistic data integration with two application scenarios. The first concerns data integration by (autonomous) mobile devices. Take, for example, an address book application on a smartphone. Such an applica-
tion stores among other things names, addresses and phone numbers of people. Other possibly mobile, autonomous devices can have an address book too, e.g., a PDA or a home PC. If a smartphone and PDA or PC are within network range, the address books on both devices can be integrated. It would be prohibitively impractical if user involvement would be required here, because this would mean that owners would constantly be interrupted by their devices, urging them to resolve conflicts and semantic uncertainties. We propose that during an ad hoc integration process, decisions on semantic uncertainties and conflicts are postponed by storing them as possibilities in the address book databases.

The main usage of an address book in a smartphone is looking up people’s phone numbers. Although the device’s database may be uncertain about the correctness of certain phone numbers, it may display the possible phone numbers as a list as usual (e.g., in decreasing order of probability). The user is likely to select the top most-likely phone number. After dialing, the person at the other end is or is not the person the user wanted to reach. Our user feedback technique allows the user to give positive or negative feedback on the result of the phone number query. The feedback is used to update the address book reducing the amount of uncertainty therein. Note that careful user-interface design can minimize additional user actions. For example, the “Hang up”-button can be given in three forms: “Person reached & Hang up”, “Incorrect & Hang up”, and “Hang up”. In some cases, the feedback need not even involve the user at all. Any interaction of the smartphone with the outside world may give rise to feedback. For example, when a “non-existing phone number” dial tone is encountered, then the smartphone may automatically give itself feedback that the phone number is incorrect.

Another scenario where probabilistic data integration with user feedback is useful is an intelligent DVD player that stores meta information on the DVD movies the user owns. The DVD player can easily extract the title of the DVD. By searching movie websites on the internet, it can integrate all information found. Semantic uncertainty easily arises however. Even with one movie website, the DVD player is confronted with conflicting information. For example, a search for a movie title “King Kong” results in information on many different movies, including the 1933, 1976, and 2005 versions of the movie (see also [KKL06]). Although some information for all three movies is the same, e.g., title, genre and maybe even language, other information is different, e.g., the year and cast. One can imagine that at one point, the user recognizes one of the actors and confirms this to the DVD player (positive feedback). With our user feedback technique, this feedback can be used to update the database in such a way that only the correct version of the movie is retained.

Both examples show how in the natural interaction with applications, users can give user feedback, both positive as well as negative. This can be used to resolve and reduce semantic uncertainty encountered during data integration which itself can be handled without user involvement.

The remainder of this paper is organized as follows. First, we give an introduction to the probabilistic XML data model and its use in data integration. In Section 3, we discuss the possible world approach underlying the various techniques we developed for probabilistic XML like integration and querying. In Section 4, we present our user feedback technique, which is also based on the possible world approach. The technique is subsequently experimentally validated in Section 5. Section 6 discusses related work. We conclude and give some directions for future research in Sections 7 and 8.

2 Probabilistic XML Integration

This section gives an introduction to probabilistic XML integration. More detailed formal definitions of the notions presented here can be found in [KKA05].

A probabilistic XML tree is a representation of uncertain XML data, i.e. data for which there is uncertainty about values and/or structure of certain elements or attributes. Two additional kinds of nodes are used to express uncertainty depicted as follows:

1. ‘\(\triangledown\)’ is a probability node,
2. ‘\(\circ\)’ is a possibility node (with associated probability), and
3. ‘\(\bullet\)’ is an ordinary XML node.

The example in figure 1 represents an address book with one (70% chance) or two person elements (30% chance). In the former case, the name of this person is for certain “John”, but there is uncertainty about the telephone number: either “1111” or “2222” with equal probability. In the latter case, the name of both persons is “John”, the first has telephone number “1111” and the second “2222”. In general, uncer-
Varying uncertainty is expressed as several possible subtrees. Dependent and independent possibilities as well as (non-)existence of elements can be expressed in this way.

Definition 1 Let \( T_{\text{fin}} \) be the set of ordered finite trees. We vary \( T \) over \( T_{\text{fin}} \). An XML document can be represented with such a tree. A probabilistic tree \( PT = (T, \text{kind}, \text{prob}) \) is a special kind of tree where the kind function assigns kinds to nodes and the prob function assigns probabilities to possibility nodes. A probabilistic XML tree is wellformed iff (1) at the successive levels in the tree, all nodes are of one kind in the order ‘\( \lor \)’, ‘o’, and ‘\( \bullet \)’; and (2) the probabilities of the child possibility nodes of one probability node always add up to one. A probabilistic XML tree is called certain if all probability nodes have only one child possibility. For simplicity, we often represent certain probabilistic trees with normal XML trees.

With probabilistic XML we capture the uncertainty arising from unattended information integration, i.e. information integration where no user is involved to solve semantic uncertainties and conflicts. For example, the probabilistic XML tree of Figure 1 could be the result of integrating the XML documents of Figure 2. We assume that the DTD specifies that a person has only one telephone number. Note that without further knowledge, it could be the case that both address books refer to two different persons called “John”. Apparently, the probability for this is estimated to be 30%. In the other case, i.e. where both address books refer to the same person named “John”, it is uncertain which of the telephone numbers is correct.

In the integration process, we defined a component called “The Oracle” that determines how likely it is that two elements refer to the same real world object. Note that our approach does not require much knowledge and intelligence in The Oracle. In the extreme case of a totally unknowledgeable Oracle, the technique falls back on the safest decisions: anything that may be possible should be stored as a possibility, however unlikely it is. As a result, uncertainty abounds and the integration result may explode in size. However, only simple knowledge rules suffice to tame this explosion to a manageable size [KKL06]. For details of the probabilistic integration technique, we refer to [KKA05].

3 Possible World Approach

Subtrees under a probability node denote local possibilities. In the address book example of Figure 1, there are two local possibilities: the phone number is either ‘1111’ or ‘2222’, and there are either one or two persons. Viewed globally and from the perspective of a database being an application’s view on the real world, this real world could either look like

- one person with name ‘John’ and phone number ‘1111’ (probability \( 0.5 \times 0.7 = 0.35 \)),
- one person with name ‘John’ and phone number ‘2222’ (probability \( 0.5 \times 0.7 = 0.35 \)), or
- two persons with name ‘John’ and respective phone numbers ‘1111’ and ‘2222’ (probability 0.3).

In other words, by choosing one local possibility for each probability node, we obtain a certain probabilistic XML tree that represents one possible appearance of the real world. These are called possible worlds. All combinations of local possibilities in a database \( PT \) give rise to the set of possible worlds \( \text{PWS}_{PT} \). Each possible world has a probability: for each \( T \in \text{PWS}_{PT} \), \( P(T \mid PT) \) is the probability of \( T \) given the probabilistic database \( PT \). It can be proven that

\[
\sum_{T \in \text{PWS}_{PT}} P(T \mid PT) = 1
\]

The meaning of a probabilistic XML database is defined by the possible worlds it represents. In fact, there may be more than one probabilistic XML tree giving rise to the same set of possible worlds. Two probabilistic XML trees \( PT_1 \) and \( PT_2 \) are equivalent iff \( \text{PWS}_{PT_1} = \text{PWS}_{PT_2} \). Among equivalent probabilistic trees, the one with the least number of nodes is called the compact representation and is the preferred way of storing the probabilistic database.

A probabilistic XML document can be queried like any other XML document. However, if the queried document is uncertain, the query result is uncertain as well. As argued in [KKA05], the semantics of a query is defined by the answers to the query in each of the possible worlds. Intuitively we say that if \( A \) is a correct view of the world and in this view the query result is \( a \), and if \( B \) happens to be a correct view of the world and in this view the query result is \( b \), then for a database being uncertain on whether \( A \) or \( B \) is correct, the logical answer on the query would be “either \( a \) or \( b \)”.

More formally, let \( Q_q \in T_{\text{fin}} \rightarrow T_{\text{fin}} \) be a query engine for a query \( q \) mapping a database to a query answer. The answer to a query on a probabilistic database can now be defined as \( Q_q(PT) = \{ Q_q(T) \mid T \in \text{PWS}_{PT} \} \). In
the XML world the query languages XPath and XQuery are used commonly. In both languages, an answer to a query is always a sequence. An uncertain answer can be represented as one probabilistic XML tree by introducing the node kind \( \text{seq} \) denoting an XML sequence. However, in this paper we represent a query answer as a set of possible query answers to simplify formulas. We use ‘∈’ to denote sequence membership.

Consider the query

```
//person[./name="John"]/phone
```

If posed to a regular XML document, the result of this query would be a sequence of phone nodes. When posed to the document in Figure 1, the result is the uncertain answer depicted in Figure 3.

The Possible World Approach presented in this section is the semantical foundation of the probabilistic data model. The actual implementation is much more efficient than just enumerating possible worlds. The operations on the data are also adapted to work on this more efficient representation.

4 User Feedback

This section describes the feedback process, what types of feedback can be given and what the effect is of feedback on the current set of possible worlds.

Note that in order to provide the underlying probabilistic XML database with feedback, the user interface will have to be altered to support this feature. The whole probabilistic architecture will work without this support, but then the amount of uncertainty in the database will never decrease.

4.1 Information Cycle

A database represents the real world. The representation ultimately comes from observations from a different entity or entities, e.g., a user or sensors. This implies that there can be a discrepancy between actual objects in the real world and the data in the database describing those objects. Figure 4 shows the entire information cycle including these observations, data integration, querying and user feedback.

Before any data is present in the information source, an external party, e.g., a person or sensor, observes an aspect of the real world and stores its interpretation thereof in the database. Information may also reach the database in an indirect way, i.e., through integration of information from other information sources. Observations and information integration may both lead to uncertainty in the database.

We represent this uncertainty with possible views of the real world, the possible worlds, that co-exist in the database.

Uncertainty can be reduced by giving feedback on query results. Because the user posing the query also observes the real world, he can determine whether certain query answers are certain correct or incorrect. By giving feedback in such cases, the database may conclude that certain possible worlds can no longer be correct and eliminate them.

The cycle of repeated observations and information integration introduces possible worlds, the cycle of repeated user feedback eliminates them. In this way, the uncertainty in the information in the database keeps reflecting the actual uncertainty on the state of affairs in the real world, and may even follow developments happening in the real world.

4.2 Types of Feedback

When a query result is returned to the user, he is already involved with the system and feedback on the validity of the query result can easily be given. Consider the query given previously asking for the phone number of persons named “John”. The answer (see Figure 3) is uncertain: either (“1111”), (“2222”), or (“1111”, “2222”). A user could readily verify these answers, for example, by calling one or both phone numbers and checking if the person on the other end of the line is named “John”. He could then indicate his findings by stating for some query results whether they are true or false in the real world. The goal of our user feedback technique is to use this information to update the information in the database accordingly, thus reducing uncertainty. We claim that a semantically correct way of doing this, is by invalidating entire possible worlds that disagree with the statement on the query result. For example, if a per-
Definition 2 Let \( S \in \overline{Q}(PT) \) be a possible query answer for some query \( q \) and probabilistic XML tree \( PT \). In XML and XPath, a query answer is always a sequence, so we assume \( S \) to be a sequence.

Negative feedback is a statement “\( a \) is false” for some \( a \in S \). The meaning of this statement is \( a \not\in Q_q(\text{RW}_{user}) \). Analogously, positive feedback is a statement “\( a \) is true” meaning \( a \in Q_q(\text{RW}_{user}) \).

\[
\overline{Q}(PT) = \{("1111"), ("2222"), ("1111", "2222")\}
\]
in our example. The positive feedback that “1111” is a correct phone number means that “1111” \( \in Q_q(\text{RW}_{user}) \), i.e. the user states that if he would ‘ask his brain’ for phone numbers of persons named “John”, “1111” is for certain among the result.

4.3 Effect of Feedback

As stated before, our approach is to invalidate, or rather eliminate, those possible worlds from the database that do not correspond with the user’s knowledge of the real world.

Definition 3 Let \( PT' \) be the result of user feedback “\( a \) is false” or “\( a \) is true” for some database \( PT \), query \( q \), and \( a \in S \), where \( S \in \overline{Q}(PT) \). \( PT' \) is defined by

\[
PWS_{PT'} = \{ T \in PWS_{PT} \mid a \not\in Q_q(T) \} \quad \text{or} \quad PWS_{PT'} = \{ T \in PWS_{PT} \mid a \not\in Q_q(T) \}, \]
respectively.

Observe that definition 3 shows that we only need to eliminate possible worlds from the database. It is never necessary to create a possible world, create or delete a local possibility, or change or delete a part of a possible world.

We have defined \( PT' \) by means of its possible worlds. Note that it is not hard to construct \( PT' \) from the set of possible worlds. Simply create a probability node with as many children as there are possible worlds. Attach each possible world, which is a certain probabilistic tree, as subtree (see Figure 5). In this way, we obtain a probabilistic tree representing exactly this set of possible worlds. Any probabilistic tree equivalent with a \( PT' \) constructed in this way, preferably the compact representation, can be used as resulting database. Actually implementing the feedback technique this way seems rather inefficient. However, in this paper we focus on the correctness of the approach, not on implementation or efficiency. We believe it is possible to construct an algorithm that applies user feedback directly on the compact representation, but this is future research.

4.4 Recalculating Probabilities

When possible worlds are removed from the database as a result of feedback, the probabilities of all remaining possible worlds have to be recalculated. Unfortunately, the databases from which the probabilistic information source originated are typically unavailable. Therefore, re-integrating sources taking feedback into account is not a viable approach. Below, we argue that the correct way of recalculation amounts to simple normalization.
Our notation \( P(T \mid PT) \) suggests that we consider the database \( PT \) as the *universe*. To emphasize this fact, we use the symbol \( U \) for the original database. Eliminating possible worlds from this universe, means constructing a (new) database \( PT' \). Let us first consider the case of a possible world \( T \) that is eliminated. Its probability \( P(T \mid PT') \) is, of course, 0. In the other case, we can calculate the probability of the possible world in the new universe using the laws of conditional probabilities as follows:

\[
P(T \mid PT') = \frac{P(T \wedge PT')}{P(PT')} = \frac{P(PT' \mid T)P(T)}{P(PT')}
\]

\( P(PT' \mid T) = 1 \), because we are considering the case that \( T \) is a member of the universe, hence the existence of the new universe given possible world \( T \) is certain. The probability of the occurrence of the new database, i.e. the new set of possible worlds, is \( P(PT') = \sum_{T \in PWS_{PT'}} P(T) \). Note that \( P(T) \) is the probability of \( T \) given our universe, hence \( P(T) = P(T \mid U) \). After substitution we finally derive

\[
P(T \mid PT') = \begin{cases} 0 & \text{if } T \text{ is eliminated} \\ \frac{P(T \mid U)}{\sum_{T' \in PWS_{PT'}} P(T') } & \text{otherwise} \end{cases}
\]

As one can observe, the new probabilities can be obtained by simply normalizing probabilities. However, the calculation given above shows that normalizing probabilities semantically fits the possible world approach.

### 4.5 Properties of Feedback

For validation purposes, we observe in this section some desirable properties of our user feedback technique. This is a kind of analytical validation. For experimental validation, we refer to Section 5.

**Property 1** Given an original database \( PT \) and a resulting database \( PT' \) after user feedback, we observe that

\( PWS_{PT'} \subseteq PWS_{PT} \)

This property follows directly from Definition 3.

**Property 2** Given an original database \( PT \) and a resulting database \( PT' \) after user feedback, we observe that

\[ \forall T \in PWS_{PT'} \cdot P(T \mid PT') \geq P(T \mid PT) \]

This property follows from the formula derived in Section 4.4. \( T \in PWS_{PT'} \) means \( T \) is not eliminated. Since \( P(T \mid PT) = P(T \mid U) \), we conclude that \( P(T \mid PT') = P(T \mid PT) \) divided by some number. Since \( \left( \sum_{T \in PWS_{PT}} P(T \mid PT) \right) = 1 \) (see Section 3) and \( PWS_{PT'} \subseteq PWS_{PT} \) (Property 1), this number is guaranteed to be smaller than or equal to one. Hence \( P(T \mid PT') \geq P(T \mid PT) \).

**Property 3** The probabilities in the new database \( PT' \) are distributed correctly, i.e.

\[
\sum_{T \in PWS_{PT'}} P(T \mid PT') = 1
\]

The property follows directly from substituting the formula from Section 4.4:

\[
\sum_{T \in PWS_{PT'}} P(T \mid PT') = \sum_{T \in PWS_{PT'}} \frac{P(T \mid U)}{\sum_{T \in PWS_{PT'}} P(T \mid U)} = \frac{\sum_{T \in PWS_{PT'}} P(T \mid U)}{\sum_{T \in PWS_{PT'}} P(T \mid U)} = 1
\]

### 4.6 Give Feedback Carefully

We mentioned earlier that a database is a representation of the real world. Although this is true, there is a need for caution, because the real world changes, hence the observation of the real world can be different from the observation at a later time. Furthermore, knowledge about the real world is always incomplete. We denote the representation of the real world as captured in the database by \( RW \). The representation of the real world as seen by the user at query time will be denoted by \( RW_{user} \).

Due to the possible (non-)overlap between real world knowledge of the database and the user, feedback to a query in terms of absolute statements should be given with caution. We will show different scenarios of mismatch in knowledge and their impact on the feedback process.

Figure 6 shows four examples of observations from the real world contained by the database and the user. The examples are restricted to a set of names of people with the same name. In this case we show all people named “John”. In each example, the left figure shows people named “John” known by the database and the right figure shows people named “John” known by the user.

Figures 6(a) and 6(b) show an ideal situation, where both the database and the user have knowledge about the same
persons. Even though their respective knowledge of these persons may differ, there is no significant mismatch between database and user and the risk of wrong feedback is minimal.

Figures 6(c) and 6(d) show the situation where the the number of people known by the database and the user is equal, but the actual data is different. In other words RW ̸= RW_{user}. Feedback about the non-existence of John_2 by this particular user could result in the deletion of all possible worlds containing John_2, while in fact that person does exist, but is just not known to the user querying the database. The user should only give such negative feedback if he is certain that it is universal, i.e. that a database containing John_2 is for certain incorrect.

Figures 6(e) and 6(f) as well as Figures 6(g) and 6(h) show situations where the number of real world objects known by the database is also different than that known by the user. In such cases, feedback on queries with aggregates are likely to result in unwanted results and should only be given with special care. Suppose a user poses the query

\[
\text{SELECT name, COUNT(*)}
\text{FROM addressbook}
\text{GROUP BY name}
\]

to see how many people with the same name he knows, i.e. are contained in the database. It could happen that the query result for a name is different than he expects and he would like to give feedback on this.

For example, in the situation of Figures 6(g) and 6(h), the query result for “John” is 2, but the user knows 3 persons named “John”. Here the user should be aware that any feedback should not only be a universal truth, but also something the database with its incomplete knowledge should know. Giving the feedback that the query result should be 3 would eliminate all possible worlds with less than or more than 3 persons, hence one could possibly end up with an empty database.

Nevertheless, feedback can be a powerful mechanism in reducing uncertainty in the database if users (or application developers) use feedback with care, i.e. only universal truths or falsehoods, and only in cases where a database with incomplete knowledge should have known it. In other words, the database should have possessed the correct information.

5 Validation

To validate our claims about the properties of the user feedback technique, we developed a prototype and carried out some experiments. Note that this paper focuses on the properties and impact of the user feedback technique, not on an efficient implementation. The experiments are designed to study the impact of feedback on the amount of uncertainty left in the database, which kind of feedback is more powerful, positive or negative, and how fast convergence can be observed. For this purpose, a rather small data set suffices. In the following, we discuss the main features of our prototype, explain in more detail the feedback process and show results from our experiments.

5.1 Prototype

Our prototype is implemented in Haskell and supports basic functionality for probabilistic data, integration, querying, and feedback. For simplicity, we did not implement full querying capabilities, but only a function called treeContainment, which checks if one XML tree is contained within another. Using treeContainment, we can support both negative and positive feedback on simple path queries.

The user feedback algorithm extracts all (probability, possible world) combinations from the database and checks whether or not each possible world meets the feedback criteria, i.e., whether the object is/is not contained in the world in case of positive/negative
Function positiveFeedback (universe, object)
    universe' = ∅
    Foreach (p, world) in universe do
        If ( treeContainment(object, world) ) then
            universe' = universe' ∪ {(p, world)}
        Endif
    done
    return normalize(universe')
EndFunction

Figure 7. Positive Feedback Algorithm

feedback. The resulting database is represented by the set of possible worlds that meet the criteria for which the probabilities are normalized. The algorithm for positive and negative feedback are rather similar, so we only present the one for positive feedback in Figure 7.

5.2 Experiments

We carried out two experiments using the integration result of the two address books shown in Figure 8. One experiment starts with an integration result obtained without using any world knowledge during data integration. For the other, one generic knowledge rule is used for data integration, hence starts with a less uncertain database. For readability, we sometimes use abbreviations for element names in our trees, e.g., ‘ln’ for ‘lastname’. We use the number of possible worlds as a measure of the amount of uncertainty in the database. Note that this number may exaggerate the amount of uncertainty as perceived by a human. For example, if a database contains three local uncertainties giving two possible values for three independent attributes, then there are eight possible worlds.

In the experiments, we give one feedback statement for each iteration and measure the uncertainty. As user feedback, we confirm the existence of persons with names Mark Hamburg, Allen King, Stan Choice, John Friend and Allen Kingship, and refute existence of people with other names like Mark King, Stan Friend, etc. So, there are 5 people we identify to exist in the real world. Note that we only make statements on existence of names. We do not make any statements on phone or room numbers. Some conclusions, such as the number of people having a certain name, cannot be deduced from this feedback.

Figures 9 and 10 show the results of both experiments. A first observation is that they confirm Property 1, because the amount of uncertainty decreases monotonously with each iteration.

Experiment 1 Because the first experiment uses no world knowledge during data integration, the starting point is a huge document containing 1815 possible worlds. Each possible world has the same, albeit small, probability.

An immediate observation is that positive feedback is stronger in reducing uncertainty than negative feedback. Since no world knowledge is used during data integration, the database contains a large variety of possibilities that can best be described as ‘utter nonsense’. Negative feedback only eliminates certain nonsense names, while positive feedback eliminates all incorrect variations related to a real world person. After 5 iterations of positive feedback, 19 possible worlds remain, which only contain persons really existing in the real world. Some uncertainty about phone or room numbers remain.

Experiment 2 As we explained in [KKL06], probabilistic information integration without any world knowledge during data integration is not realistic in practice. Therefore, we repeat our experiment for an integrated information source for which we used one simple generic knowledge rule during integration to exclude the most nonsensical possibilities. The rule states that person elements can only refer to the same real world person if at least one of the attribute values is equal. The initial database for this experiment contains 39 possible worlds.

Again, positive feedback is stronger. After 2 iterations, there is even no improvement anymore with negative feedback. With positive feedback, the only uncertainty that the system could not resolve, is about the phone number of “Mark Hamburg”. There could exist only one person with that name and one of the phone numbers is incorrect, or there exist in fact two persons with that name and both phone numbers are correct. This cannot be deduced from feedback on names only.
5.3 Evaluation of the results

In both experiments, the number of possible worlds decreases and no correct information is deleted. Furthermore, only a few iterations are needed to arrive at only a few possible worlds, for which it is obvious that the system could not get rid of based on the restricted feedback we gave. In other words, convergence to the truth may not happen in certain circumstances, but convergence to some amount of uncertainty is quick. The second experiment also showed that using a very simple world knowledge at integration time, the number of possible worlds decreased drastically. Note that the order of feedback statements is irrelevant. Any order arrives at the same set of possible worlds.

6 Related Work

A probabilistic database is not a new idea, see for example [FKL97], but in recent years attention grew considerably. Originally, work concentrated on relational databases, but in [KKA05] we argue that XML can be made to express uncertainty in a more natural way. Other probabilistic XML databases are, for example, PXML [HGS03], ProTDB [NJ02] and more recently Trio [BSHW06].

In this paper we suggest that one of the application areas where probabilistic data can be useful is data integration. There is a large body of related work on information integration. [DH05] provides a nice survey. It is useful to distinguish schema matching and integration from data integration. In this paper, we focus on data integration as an application of probabilistic XML, hence we presume that schema matching and integration has already been done. [RB01] is a good survey on schema matching techniques.

To our knowledge, there is no previous work on user feedback in probabilistic XML. However, in information retrieval the topic of user feedback is a known research area.

In information and multimedia retrieval, relevance feedback is a well-known solution for allowing a user to give feedback on a query result. The result of a query in this application area is a sequence of documents or multimedia objects ranked on relevancy with respect to the query. The user gives feedback by indicating which documents or multimedia objects he perceives as relevant to his information need. The system uses this feedback to reformulate the query based on the features of the selected documents or multimedia objects. After feedback, the revised query is executed and, if all goes well, the top-most documents or multimedia objects better fulfill the user’s information need. A notable difference with our kind of user feedback is that
here the feedback is used to reformulate the query, not to change the document or multimedia collection. A notable similarity is that both positive and negative feedback is considered and remarkably positive feedback appears to work better as well. We cite [BYRN99] as a reputable textbook containing a number of references to related work of relevance feedback.

7 Conclusions

Manual information integration is a labor intensive process, unfortunately automatic integration is very error prone. The proposed feedback mechanism combines maximal influence from the user in normal integration with the unattendedness from the automatic integration by postponing user involvement to query time. The user involvement is restricted to giving positive and negative feedback on query results, which can be integrated in the natural interaction of the user with applications. The result is a mechanism that provides a simple means to integrating information sources without losing information, by first storing all possibilities related to conflicts and semantic uncertainty, and second using user feedback for reducing uncertainty.

Experiments show that using the feedback mechanism, an integrated information source (partially) converges to a correct representation of the real world, provided that the user provides correct feedback considering the fact that he himself as well as the information source have an incomplete view on the real world. Therefore, the integrated information source converges to something that possibly still contains some uncertainty. Furthermore, experiments show as well that giving positive feedback is usually more effective in reducing uncertainty than negative feedback.

8 Future Research

Although we used our prototype to experiment on a small dataset, we plan to also perform experiments on much larger real-life datasets. We plan to integrate several information sources containing movie data as described in [KKL06].

The essence of our techniques is based on the possible world approach. Therefore, all operations are described in terms of actions on possible worlds. For smaller datasets this is possible, but with growing datasets working on the set of possible worlds is infeasible due to the amount of data involved. We have defined the concept of compact representation, so an operation working on a set of possible worlds can also be redefined to work on the compact representation. We intend to develop efficient algorithms for these operations.

Besides feedback, an alternative solution to the data explosion problem would be to introduce ignorance in the uncertainty model [BGMP90]. With ignorance it is no longer necessary to always enumerate all possibilities, because it allows you to forget information. The datamodel would become more expressive, but all techniques already developed to be used with our probabilistic tree would have to be re-evaluated and possibly changed to support this ignorance.

In this paper, we have shown that the set of possible worlds converges and that the amount of false information decreases after feedback. However, the resulting information source may never become entirely certain. One remaining research question is, which category of possible worlds remains. This could help in optimizing the feedback process and perhaps even let the set of possible worlds converge to a certain XML tree.

References


