Duplicate Detection in Probabilistic Data

Fabian Panse #1, Maurice van Keulen *2, Ander de Keijzer *3, Norbert Ritter #4

# Computer Science Department, University of Hamburg
Vogt-Koelln Straße 33, 22527 Hamburg, Germany
1 panse@informatik.uni-hamburg.de
2  ritter@informatik.uni-hamburg.de

* Faculty of EEMCS, University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
3 m.vankeulen@utwente.nl
3 a.dekeijzer@utwente.nl

Abstract—Collected data often contains uncertainties. Probabilistic databases have been proposed to manage uncertain data. To combine data from multiple autonomous probabilistic databases, an integration of probabilistic data has to be performed. Until now, however, data integration approaches have focused on the integration of certain source data (relational or XML). There is no work on the integration of uncertain (esp. probabilistic) source data so far. In this paper, we present a first step towards a concise consolidation of probabilistic data. We focus on duplicate detection as a representative and essential step in an integration process. We present techniques for identifying multiple probabilistic representations of the same real-world entities. Furthermore, for increasing the efficiency of the duplicate detection process we introduce search space reduction methods adapted to probabilistic data.

I. INTRODUCTION

In a large number of application areas (e.g., astronomy [1]), the demand for storing uncertain data grows increasingly from year to year. As a consequence, in the last decades several probabilistic data models have been proposed (e.g., [2], [3], [4], [5], [6]) and recently several probabilistic database prototypes have been designed (e.g., [7], [8], [9]).

In current research on data integration, probabilistic data models are only considered for handling uncertainty in an integration of certain source data (e.g., relational [10], [11] or XML [12]). Integration of uncertain (esp. probabilistic) source data has not been considered so far. However, to consolidate multiple probabilistic databases to a single one, for example for unifying data produced by different space telescopes, an integration of probabilistic source data is necessary.

In general, an integration process mainly consists of four steps: (a) schema matching [13] and (b) schema mapping [14] to overcome schema and data heterogeneity; (c) duplicate detection [15] (also known as record linkage [16]) and (d) data fusion [17] to reconcile data about the same real-world entities (in the literature, the composition of the last two steps is also known as entity resolution [18] or the merge/purge problem [19]). In this paper, we focus on duplicate detection as a representative step in the data integration process and show how to adapt existing techniques to probabilistic data.

The remainder of this paper is structured as follows. First we present related work (Section II). In Section III, we examine current techniques of duplicate detection in certain data. Then we introduce duplicate detection for probabilistic databases in Section IV. In Section V, we identify search space reduction techniques for probabilistic data making the duplicate detection process more feasible. Finally, Section VI concludes the paper and gives an outlook on future research.

II. RELATED WORK

In general, probability theory is already applied in methods for duplicate detection (e.g., decision models), but current approaches only consider certain relational ([18], [16], [19]) or XML data [20]. Probabilistic source data is not considered in these works. On the other hand, many techniques that focus on data preparation [21] and verification [22] as well as fundamental concepts of decision model techniques [22] can be adopted for duplicate detection in probabilistic data. Furthermore, existing comparison functions [15] can be incorporated into techniques for comparing probabilistic values.

There are several approaches that explicitly handle and produce probabilistic data in schema integration, duplicate detection and data fusion. Handling the uncertainty in schema integration requires probabilistic schema mappings [11], [23]. Van Keulen and De Keijzer ([6], [24], [12]) use a semi-structured probabilistic model to handle ambiguities arising during deduplication in XML data. Tseng [10] already used probabilistic values in order to resolve conflicts between two or more certain relational values. None of the studies, however, allows probabilistic data as source data.

III. FUNDAMENTALS OF DUPLICATE DETECTION

The data sets to be integrated may contain data on the same real-world entities. Often it is even the purpose of integration: to combine data on these entities. In order to integrate two or more data sets in a meaningful way, it is necessary to identify representations belonging to the same real-world entity. Therefore, duplicate detection is an important component in an integration process. Due to deficiencies in data collection, data modeling or data management, real-life data is often incorrect and/or incomplete. This principally hinders duplicate detection. Therefore, duplicate detection techniques have to be

Abstract—Collected data often contains uncertainties. Probabilistic databases have been proposed to manage uncertain data. To combine data from multiple autonomous probabilistic databases, an integration of probabilistic data has to be performed. Until now, however, data integration approaches have focused on the integration of certain source data (relational or XML). There is no work on the integration of uncertain (esp. probabilistic) source data so far. In this paper, we present a first step towards a concise consolidation of probabilistic data. We focus on duplicate detection as a representative and essential step in an integration process. We present techniques for identifying multiple probabilistic representations of the same real-world entities. Furthermore, for increasing the efficiency of the duplicate detection process we introduce search space reduction methods adapted to probabilistic data.

I. INTRODUCTION

In a large number of application areas (e.g., astronomy [1]), the demand for storing uncertain data grows increasingly from year to year. As a consequence, in the last decades several probabilistic data models have been proposed (e.g., [2], [3], [4], [5], [6]) and recently several probabilistic database prototypes have been designed (e.g., [7], [8], [9]).

In current research on data integration, probabilistic data models are only considered for handling uncertainty in an integration of certain source data (e.g., relational [10], [11] or XML [12]). Integration of uncertain (esp. probabilistic) source data has not been considered so far. However, to consolidate multiple probabilistic databases to a single one, for example for unifying data produced by different space telescopes, an integration of probabilistic source data is necessary.

In general, an integration process mainly consists of four steps: (a) schema matching [13] and (b) schema mapping [14] to overcome schema and data heterogeneity; (c) duplicate detection [15] (also known as record linkage [16]) and (d) data fusion [17] to reconcile data about the same real-world entities (in the literature, the composition of the last two steps is also known as entity resolution [18] or the merge/purge problem [19]). In this paper, we focus on duplicate detection as a representative step in the data integration process and show how to adapt existing techniques to probabilistic data.

The remainder of this paper is structured as follows. First we present related work (Section II). In Section III, we examine current techniques of duplicate detection in certain data. Then we introduce duplicate detection for probabilistic databases in Section IV. In Section V, we identify search space reduction techniques for probabilistic data making the duplicate detection process more feasible. Finally, Section VI concludes the paper and gives an outlook on future research.

II. RELATED WORK

In general, probability theory is already applied in methods for duplicate detection (e.g., decision models), but current approaches only consider certain relational ([18], [16], [19]) or XML data [20]. Probabilistic source data is not considered in these works. On the other hand, many techniques that focus on data preparation [21] and verification [22] as well as fundamental concepts of decision model techniques [22] can be adopted for duplicate detection in probabilistic data. Furthermore, existing comparison functions [15] can be incorporated into techniques for comparing probabilistic values.

There are several approaches that explicitly handle and produce probabilistic data in schema integration, duplicate detection and data fusion. Handling the uncertainty in schema integration requires probabilistic schema mappings [11], [23]. Van Keulen and De Keijzer ([6], [24], [12]) use a semi-structured probabilistic model to handle ambiguities arising during deduplication in XML data. Tseng [10] already used probabilistic values in order to resolve conflicts between two or more certain relational values. None of the studies, however, allows probabilistic data as source data.

III. FUNDAMENTALS OF DUPLICATE DETECTION

The data sets to be integrated may contain data on the same real-world entities. Often it is even the purpose of integration: to combine data on these entities. In order to integrate two or more data sets in a meaningful way, it is necessary to identify representations belonging to the same real-world entity. Therefore, duplicate detection is an important component in an integration process. Due to deficiencies in data collection, data modeling or data management, real-life data is often incorrect and/or incomplete. This principally hinders duplicate detection. Therefore, duplicate detection techniques have to be
designed for properly handling dissimilarities due to missing data, typos, data obsolescence or misspellings.

In general, duplicate detection consists of five steps [22]:

A. Data Preparation

Data is standardized (e.g., unification of conventions and units) and cleaned (elimination of easy to recognize errors) to obtain a homogeneous representation of all source data [21].

B. Search Space Reduction

Since a comparison of all combinations of tuples is mostly too inefficient, the search space is usually reduced using heuristic methods such as the sorted neighborhood method, pruning or blocking [22].

C. Attribute Value Matching

Similarity of tuples is usually based on the similarity of their corresponding attribute values. Despite data preparation, syntactic as well as semantic irregularities remain. Thus, attribute value similarity is quantified by syntactic (e.g., n-grams, edit- or jaro distance [15]) and semantic (e.g., glossaries or ontologies) means. From comparing two tuples, we obtain a comparison vector \( \vec{c} = [c_1, \ldots, c_n] \), where \( c_i \) represents the similarity of the values from the \( i \)th attribute.

D. Decision Model

The comparison vector is input to a decision model which determines to which set a tuple pair \((t_1, t_2)\) is assigned: matching tuples \((M)\), unmatching tuples \((U)\) or possibly matching tuples \((P)\). In the following, the decision’s result is stored in the matching value \( \eta(t_1, t_2) \in \{m, p, u\} \), where \( m \) represents the case that \((t_1, t_2)\) is assigned to \( M \) (resp. to \( P \) or \( U \)).

The most common decision models are based on domain knowledge or probability theory:

Knowledge-based techniques. In knowledge-based approaches for duplicate detection [22], domain experts define identification rules. Identification rules specify conditions when two tuples are considered duplicates with a given confidence (certainty factor). An example of such a rule is shown in Figure 1. This rule defines that two tuples are duplicates with a certainty of 80%, if the similarities of their names and jobs are greater than the corresponding thresholds. Ultimately, if the resulting certainty is greater than a third, user-defined threshold separating \( M \) and \( U \), the tuple pair is considered to be a duplicate (the set \( P \) is usually not considered in works on these techniques).

Probabilistic techniques. In the theory of fellegri and sunter ([16], [22]), two conditional probabilities \( m(\vec{c}) \) (m-probability) and \( u(\vec{c}) \) (u-probability) are defined for each tuple pair \((t_1, t_2)\).

\[
\begin{align*}
    m(\vec{c}) &= P(\vec{c} \mid (t_1, t_2) \in M) \quad (1) \\
    u(\vec{c}) &= P(\vec{c} \mid (t_1, t_2) \in U) \quad (2)
\end{align*}
\]

Based on the matching weight \( R = m(\vec{c})/u(\vec{c}) \) and the thresholds \( T_\mu \) and \( T_\lambda \), the tuple pair \((t_1, t_2)\) is considered to be a match, if \( R > T_\mu \) or a non-match, if \( R < T_\lambda \) (see Figure 2). Otherwise, the tuples are a possible match and clerical reviews are required. For computing or estimating \( m \)- and \( u \)-probabilities as well as the two thresholds \( T_\mu \) and \( T_\lambda \), several methods (with or without labeled training data) have been proposed in the literature ([25], [26], [27], [28]).

In general, the decision whether a tuple pair \((t_1, t_2)\) is a match or not, can be decomposed into two steps (see Figure 3).

In the first step, a single similarity degree \( \mathit{sim}(t_1, t_2) \) is determined by a combination function:

\[
\varphi : [0,1]^n \to \mathbb{R} \quad \mathit{sim}(t_1, t_2) = \varphi(\vec{c}) \quad (3)
\]

The resulting degree is normalized, if a knowledge-based technique is used (certainty factor) and non-normalized if a probabilistic technique is applied (matching weight). In a second step, based on \( \mathit{sim}(t_1, t_2) \) the tuple pair is assigned to one of the sets \( M \), \( P \) or \( U \) by using one or two thresholds (depending on the support for a set of possible matches).

E. Verification

The effectiveness of the applied identification is checked in terms of recall, precision, false negative percentage, false positive percentage and \( F_1 \)-measure [22]. If the effectiveness is not satisfactory, duplicate detection is repeated with other, better suitable thresholds or methods (e.g., other comparison functions or decision models).
IV. Duplicate Detection in Probabilistic Data

Theoretically, a probabilistic database is defined as $PDB = (W, P)$ where $W = \{I_1, \ldots, I_n\}$ is the set of possible worlds and $P : W \rightarrow (0, 1]$, $\sum_{I \in W} P(I) = 1$ is the probability distribution over these worlds. Because the data of individual worlds often considerably overlaps and it is sometimes even impossible to store them separately (e.g., if $|W| \rightarrow \infty$) a succinct representation has to be used.

In probabilistic relational models, uncertainty is modeled on two levels: (a) each tuple $t$ is assigned with a probability $p(t) \in (0, 1]$ denoting the likelihood that $t$ belongs to the corresponding relation (tuple level), and (b) alternatives for attribute values are given (attribute value level).

In earlier approaches, alternatives of different attribute values are considered to be independent (e.g., [3]). In these models, each attribute value can be considered as a separate random variable with its own probability distribution. Newer models like Trio [7], [29], [30] or MayBMS [8], [31] support dependencies by introducing new concepts like Trio’s x-tuple and MayBMS’s U-relation. For ease of presentation, we focus on duplicate detection in probabilistic data models without dependencies first, before considering x-tuples.

In general, tuple membership in a relation (uncertainty on tuple level) results from the application context. For example, a person can be stored in two different relations: one storing adults, the other storing people having a job. If we assume that the considered person is certainly 34 years old and jobless with a confidence of 90%, then the probability that a tuple $t_1$ representing this person belongs to the first relation is $p(t_1) = 1.0$, but the probability that a corresponding tuple $t_2$ belongs to the second relation is only $p(t_2) = 0.1$. Note that both tuples represent the same person despite the significant difference in probabilities. This illustrates that not tuple membership but only uncertainty on attribute value level should influence the duplicate detection process (see Section IV-B).

A. Duplicate detection in models without dependencies

Consider the two probabilistic relations to be integrated, $R_1$ and $R_2$ as shown in Figure 4. Both relations contain uncertainty on tuple level and attribute value level. Note that the person represented by tuple $t_{11}$ is jobless with a probability of 10%. In the following, this notion of non-existence (meaning that for the corresponding object such a property does not exist) is denoted by $\perp$.

Since no dependencies exist, similarity can still be determined on an attribute-by-attribute basis. Two non-existent values refer to the same fact of the real-world, namely that the corresponding property of the considered objects does not exist for both of them. A non-existent value, however, is definitely not similar with any existing one. Thus, we define $sim(\perp, \perp) = 1$ and $sim(a, \perp) = sim(\perp, a) = 0$ ($a \neq \perp$).

Assuming error-free data, the similarity of two uncertain attribute values $a_1$ and $a_2$ each defined in the domain $D$ ($D = \{D \cup \perp\}$) can be defined as the probability that both values are equal:

$$sim(a_1, a_2) = P(a_1 = a_2) = \sum_{d \in D} P(a_1 = d, a_2 = d) \quad (4)$$

In erroneous data, the similarity of domain elements has to be additionally taken into account:

$$sim(a_1, a_2) = \sum_{d_1 \in D} \sum_{d_2 \in D} P(a_1 = d_1, a_2 = d_2) \cdot sim(d_1, d_2) \quad (5)$$

For instance, the similarity of $t_{11}$.name and $t_{22}$.name is either $sim$($\text{Name}(Tim)$, $\text{Name}(Tim)$) = 1 (with probability 0.7) or $sim$($\text{Name}(Tim)$, $\text{Name}(Kim)$) = $\alpha$ (with probability 0.3), where $\alpha$ depends on the chosen comparison function. For example, if we take the normalized hamming distance, $\alpha = 2/3$ and hence the similarity of both attribute values results in $sim(t_{11}$.name, $t_{22}$.name) = 0.9. By using the same distance, the similarities $sim$($\text{machinist}$, $\text{mechanic}$) = 5/9 and hence $sim(t_{11}$.job, $t_{22}$.job) = 0.2 + 0.7 \cdot 5/9 = 0.59$ result.

Common decision models can be used without any adaption, because uncertainty is handled on the attribute value level and matching invariably results in a comparison vector $\bar{c}$. For example, if we use the simple combination function

$$\varphi(\bar{c}) = 0.8 \cdot c_1 + 0.2 \cdot c_2$$

for calculating tuple similarity, the similarity of $t_{11}$ and $t_{22}$ results in $sim(t_{11}, t_{22}) = 0.8 \cdot 0.9 + 0.2 \cdot 0.59 = 0.838$.

B. Duplicate detection in models with x-tuples

To model dependencies between attribute values, the concept of x-tuples is introduced in the ULDB model of Trio [29], [30]. An x-tuple $t$ consists of one or more alternative tuples ($t^1, \ldots, t^n$) which are mutually exclusive. The ULDB model does not support an infinite number of alternatives (e.g., uncertainty in a continuous domain). In these cases, and to avoid high numbers of alternatives, a probability distribution can sometimes still be associated with the attribute value. For example the value ‘mu*’ (see $t_{22}$.job) represents a uniform distribution over all possible jobs starting with the characters ‘mu’ (e.g., musician). Maybe x-tuples (tuples for which non-existence is possible, i.e., for which the probability sum of the alternatives is smaller than 1) are indicated by ‘?’. Relations containing one or more x-tuples are called x-relations. For demonstrating duplicate detection in data models supporting
the x-tuple concept, we consider a consolidation of the two x-relations $\mathcal{R}_3$ and $\mathcal{R}_4$ of Figure 5.

<table>
<thead>
<tr>
<th>name</th>
<th>job</th>
<th>$p(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>pilot</td>
<td>0.7</td>
</tr>
<tr>
<td>Johan</td>
<td>musician</td>
<td>0.3</td>
</tr>
<tr>
<td>Tim</td>
<td>mechanic</td>
<td>0.3</td>
</tr>
<tr>
<td>Jim</td>
<td>mechanic</td>
<td>0.2</td>
</tr>
<tr>
<td>Jim</td>
<td>baker</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>job</th>
<th>$p(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>pilot</td>
<td>0.8</td>
</tr>
<tr>
<td>Johan</td>
<td>pianist</td>
<td>0.2</td>
</tr>
<tr>
<td>Tom</td>
<td>mechanic</td>
<td>0.8</td>
</tr>
<tr>
<td>John</td>
<td>pilot</td>
<td>0.2</td>
</tr>
<tr>
<td>Sean</td>
<td>pilot</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Fig. 5. X-relations $\mathcal{R}_3$ (left) and $\mathcal{R}_4$ (right).

Principally, we derive the similarity of two x-tuples $t_1 = \{i^t_1, \ldots, i^t_k\}$ and $t_2 = \{i^t_2, \ldots, i^t_l\}$ from the similarity of their alternative tuples. Therefore, in the attribute value matching step, the attribute values of all alternative tuples of $t_1$ and all alternative tuples of $t_2$ are pairwise compared. Since individual attribute values (e.g., $t_{321}^2$:job) can be uncertain, we use the formulas of Section IV-A. In this way, instead one single vector $\vec{c}, k \times l$ comparison vectors are obtained. Therefore, decision models for assigning the pair $(t_1, t_2)$ to one of the sets $M, P$ or $U$ need to be adapted.

We define two approaches (see Figure 6). For each approach, the input consists of the considered x-tuple pair $(t_1, t_2)$ and a comparison matrix containing the comparison vector of each alternative tuple pair $(t^i_1, t^i_2)$. In the first approach (Figure 6, left side), the similarity of the x-tuples is based on the similarity of their alternative tuples ($\varphi : \mathbb{R}^{k \times l} \rightarrow \mathbb{R}$). In the second approach (Figure 6, right side), it is derived from their matching results ($\varphi : \{m, p, u\}^{k \times l} \rightarrow \mathbb{R}$).

**similarity-based derivation.** In more detail, the first, more intuitive approach is based on the similarity vector $s(t_1, t_2)$ containing the similarity of each alternative tuple pair $(t^i_1, t^i_2)$ which is determined by $\varphi(\vec{c}_{ij})$ (Step 1). The final similarity $\text{sim}(t_1, t_2)$ results from a derivation function $\varphi(s(t_1, t_2))$ (Step 2). Ultimately, the x-tuple pair is classified into $\{M, P\}$ or $\{M, P, U\}$ by comparing $\text{sim}(t_1, t_2)$ with one or two thresholds (Step 3). Since the similarity of two x-tuples is directly derived from the similarities of their alternative tuples, this approach is denoted as similarity-based derivation.

One adequate derivation is to calculate the expected value of the alternative tuple similarities. Since tuple membership is not relevant for duplicate detection, the probability of each alternative tuple $t^i$ has to be normalized w.r.t. the probability of the corresponding x-tuple ($p(t^i)/p(t)$), where $p(t) = \sum_{j \in [1, n]} p(t^j)$. Resulting from this normalization (also known as conditioning [32] or scaling [33]) the similarity of the two x-tuples $t_1$ and $t_2$ is defined as the conditional expectation $\varphi(s(t_1, t_2)) = E(\text{sim}(t^i_1, t^i_2) | B)$, where $B$ is the event that both tuples belong to their corresponding relation, and hence results in:

$$\text{sim}(t_1, t_2) = \frac{\sum_{i \in [1, k]} \sum_{j \in [1, l]} p(t^i_1) p(t^i_2)}{p(t_1) p(t_2)} \cdot \text{sim}(t^i_1, t^i_2) \quad (6)$$

Note that equations 5 and 6 are equivalent to the expected value of the corresponding similarity over all possible worlds containing the considered tuples.

As an example, we consider the two x-tuples $t_{32}$ and $t_{42}$. With respect to these two x-tuples there exist the eight possible worlds $\{I_1, I_2, \ldots, I_8\}$ shown in Figure 7. Both tuples should belong to their corresponding relation (event $B$). The database conditioned with $B$ is obtained by removing the possible worlds $\{I_4, I_5, I_6, I_7, I_8\}$. The probabilities of the three remaining worlds have to be renormalized to have again sum up to 1. From these renormalizations the conditional probabilities $P(I_1 | B), P(I_2 | B)$ and $P(I_3 | B)$ result from dividing the original probabilities by

$$P(B) = P(I_1) + P(I_2) + P(I_3) = (p(t^i_{32}) + p(t^i_{32}) + p(t^i_{32})) \cdot p(t_{42}) = p(t^i_{32}) = 0.72$$

The similarity of $t_{32}$ and $t_{42}$ in the possible world $I_1$ is the similarity of the two alternative tuples $t^i_{32}$ and $t^i_{42}$. In world $I_2$ (resp. $I_3$), this similarity is equal to the similarity $\text{sim}(t^i_{32}, t^i_{42})$.
job Tim mechanic

job Tim mechanic

job Tom mechanic

job Jim mechanic

(resp. \( \text{sim}(t_{32}^1, t_{42}^2) \)). As a consequence, the expected similarity \( E(\text{sim}(t_{32}^1, t_{42}^2)|B) \) and hence the similarity of both tuples result in:

\[
\text{sim}(t_{32}, t_{42}) = \frac{P(I_1)/P(B) \cdot \text{sim}(t_{32}^1, t_{42}^2) + P(I_2)/P(B) \cdot \text{sim}(t_{32}^2, t_{42}^2) + P(I_3)/P(B) \cdot \text{sim}(t_{32}^3, t_{42}^2)}{P(t_{32})} . \]

Unfortunately, if the values resulting from Step 1 are not normalized, the expected value \( E(\text{sim}(t_1, t_2)|B) \) can become unrepresentative. For example, if the alternative tuples \( t_1^1 \) and \( t_2^1 \) are similar to a large extent (\( \phi(c_3) \to \infty \)), the similarity \( \text{sim}(t_1, t_2) \) becomes infinite, too, independent from the probability of these alternatives. As a consequence, this approach is more fitting for knowledge-based than for probabilistic techniques.

decision-based derivation. In the second approach, after calculating the similarity of all alternative tuple pairs (Step 1.1), each of these pairs is classified into \( \{M, P, U\} \) (Step 1.2). From the resulting matching vector \( \vec{m} = \{m, p, u\}^{k \times l} \), the similarity of the corresponding x-tuples is derived (Step 2) and the tuple pair is assigned to one of the three sets \( M \) and \( U \) (Step 3). In this approach, the similarity of two x-tuples is derived from the decisions whether their alternative tuple pairs are duplicates or not. As a consequence, this approach is denoted as decision-based derivation.

The derivation function \( \theta \) of Step 2 can be based on probability theory. For example, by defining the tuple similarity \( \text{sim}(t_1, t_2) \) as a kind of matching weight:

\[
\text{sim}(t_1, t_2) = P(m)/P(u) \quad (7)
\]

where the two probabilities \( P(m) \) and \( P(u) \) are defined as:

\[
P(m) = \sum_{(t_1', t_2') \in M} \frac{p(t_1') \cdot p(t_2')}{p(t_1) \cdot p(t_2)} \quad (8)
\]

\[
P(u) = \sum_{(t_1', t_2') \in U} \frac{p(t_1') \cdot p(t_2')}{p(t_1) \cdot p(t_2)} \quad (9)
\]

\( P(m) \) is the overall probability of all possible worlds in which both tuples are determined to be a match. In contrast, \( P(u) \) is the overall probability of all possible worlds in which both tuples are determined to be a non-match. Thus, this derivation is based on the idea that the greater the difference between the probabilities of the alternative tuple pairs determined as a match, and the probabilities of the alternative tuple pairs determined as a non-match (and hence the difference between the overall probabilities of the corresponding possible worlds), the greater is the similarity of both tuples.

As an example, we once more consider the two x-tuples \( t_{32} \) and \( t_{42} \) and hence the possible worlds \( I_1, I_2 \) and \( I_3 \). If we define the two thresholds \( T_3 = 0.4 \) and \( T_2 = 0.7 \), in world \( I_1 \) both tuples are declared as a match. In contrast in world \( I_2 \) both tuples are determined to be a non-match. Moreover, in world \( I_3 \) the tuple pair is assigned to the set of possible matches. As a consequence, the probability \( P(m) \) is equal to the conditional probability \( P(I_1|B) = 3/9 \) and \( P(u) \) is equal to \( P(I_3|B) = 4/9 \). Accordingly, the similarity of \( t_{32} \) and \( t_{42} \) results in \( \text{sim}(t_{32}, t_{42}) = (3/9)/(4/9) = 0.75 \) (note that this value is non-normalized).

Since in this approach the similarity of two x-tuples is based on values defined in the discrete domain \( \{m, p, u\} \), the x-tuple similarity is naturally more imprecise than in a similarity-based derivation. In contrast, in spite of unnormalized results of Step 1, cases of total unrepresentative similarity values can be avoided.
In summary, a similarity-based derivation is more suitable for knowledge-based techniques (for example by calculating the expected certainty in Step 2) and a decision-based derivation is more adequate for probabilistic techniques.

Even though we only present one derivation for each approach in this paper, further adequate derivation functions are possible. For example, another decision-based derivation results by defining \( \theta \) as the expected matching result of the alternative tuple pairs \( E(\eta(t_1^1, t_2^1)|B) \), where each matching result is considered as one of the following numbers \( \{m = 2, p = 1, u = 0\} \).

V. SEARCH SPACE REDUCTION

As already mentioned in Section III, duplicate detection requires the comparison of all tuples which each other. With growing data size, this quickly becomes inefficient and perhaps even prohibitive. Therefore, the search space has to be reduced in a way that has a low risk of losing matches, for example by applying heuristic methods such as the sorted neighborhood method or blocking. In both methods a key has to be defined. In probabilistic databases, this is especially difficult, if the defined key includes uncertain attributes. For instance, in our examples a key could contain the first three characters of the name value and the first two characters of the job value. Unfortunately, for tuple \( t_{22} \) it is not clear which of the possible names has to be used for creating the key value. As a consequence, these heuristics need to be adapted to probabilistic data.

A. Sorted Neighborhood Method

In the sorted neighborhood method ([19], [22]), the key is used for tuple sorting. In probabilistic databases key values often have to be created from probabilistic data. There are basically four approaches to handle this problem. The first three attempts to obtain certain key values. The fourth adapts the sorted neighborhood method to uncertain key values.

1) Multi-Pass over Possible Worlds: A first intuitive approach is a multi-pass approach. In each pass the key values are created for exact one possible world. In this way, the key values are always certain and the sorted neighborhood method can be applied as usual. Note, since tuple membership should not influence the duplicate detection process and each tuple has to be assigned to a key value, only possible worlds containing all tuples have to be considered.

2) Creation of Certain Key Values: Alternatively, certain key values can be obtained by unifying tuple alternatives to a single one before applying the key creation function. In general, conflict resolution strategies known from techniques for the fusion of certain data [17] can be used. For example, according to a metadata based deciding strategy the most probable alternative can be chosen. This results in a sorting of \( R_{34} \) as shown in Figure 10.

Note, choosing the most probable alternatives for key value creation is equivalent to take the most probable world. Thus, the set of matchings resulting from this strategy is always a subset of the matchings resulting from the multi-pass approach presented previously.

3) Sorting Alternatives: Moreover, key values for all (or the most probable) tuple alternatives can be created. In this way, each tuple can have multiple key values. Finally, the alternatives’ key values can be sorted while keeping references to the tuples they belong to (see Figure 11). As a consequence, each tuple appears in the sorted relation for multiple times (e.g., \( t_{32} \) appears for three times). Obviously, matching a tuple with itself is meaningless. Therefore, if two neighboring key values...
are referencing to the same tuple, one of this values can be omitted (e.g., see the first two entries of the sorted relation).

<table>
<thead>
<tr>
<th>key value</th>
<th>tuple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johpi</td>
<td>t_{31}</td>
</tr>
<tr>
<td>Johmu</td>
<td>t_{32}</td>
</tr>
<tr>
<td>Timme</td>
<td>t_{32}</td>
</tr>
<tr>
<td>Jimme</td>
<td>t_{32}</td>
</tr>
<tr>
<td>Jimba</td>
<td>t_{41}</td>
</tr>
<tr>
<td>Johpi</td>
<td>t_{41}</td>
</tr>
<tr>
<td>Tomme</td>
<td>t_{42}</td>
</tr>
<tr>
<td>Joh</td>
<td>t_{43}</td>
</tr>
<tr>
<td>Seapi</td>
<td>t_{43}</td>
</tr>
</tbody>
</table>

Fig. 11. Sorting alternatives

This approach may result, however, in multiple matchings of the same tuple pair. This can be avoided by storing already executed matchings (see matrix in Figure 12).

As an example, assuming a window size of 2, from the ten possible x-tuple matchings of \( \mathcal{R}_{34} \) (intra- as well as intersource) five matchings are applied (each for exact one time): \((t_{32}, t_{43})\) (entries 1 and 3), \((t_{43}, t_{31})\) (entries 3 and 4), \((t_{31}, t_{41})\) (entries 4 and 6), \((t_{41}, t_{43})\) (entries 6 and 7) and \((t_{32}, t_{42})\) (entries 8 and 9).

![Fig. 12. Matrix for storing already executed matchings](image)

4) Handling of Uncertain Key Values: Another and w.r.t. effectiveness more promising approach is to allow uncertain key values and to sort the tuples by using a ranking function as proposed for probabilistic databases (e.g., [34], [35], [36], [37]). In general, a probabilistic relation can be ranked with a complexity of \( O(n \cdot \log n) \) (see the ranking function \( PRF^c \) in [37]). Thus, the complexity of this approach is equal to the complexity of sorting tuples in relations with certain data [22]. As an illustration, sorting based on the probabilistic key values of relation \( \mathcal{R}_{34} \) created by using the key defined above is shown in Figure 13. Note that \( t_{41} \) has a certain key value despite of having two alternative tuples.

![Fig. 13. Sorting based on the uncertain key values of relation \( \mathcal{R}_{34} \)](image)

B. Blocking

With blocking [22], the considered tuples are partitioned into mutually exclusive blocks. Finally, only tuples in one block are compared with each other. The partition can be realized by choosing a blocking key and grouping into a block all tuples that have the same key value. As for the sorted neighborhood method, a multi-pass approach over all possible worlds is most often not efficient. However, a multi-pass over some finely chosen worlds seems to be an option. Furthermore, as known from the sorted neighborhood method, conflict resolution strategies can be used to produce certain key values. In this case, blocking can be performed as usual. Handlings for uncertain key values can be based on clustering techniques for uncertain data (e.g., [38], [39], [40]).

Moreover, similar to the approach of sorting alternatives an x-tuple can be inserted into multiple blocks by creating a key for each alternative. An example for blocking with alternative key values is shown in Figure 14. The tuples are partitioned into six blocks by using a key consist of the first character of the name and the first character of the job. If an x-tuple is allocated to a single block for multiple times (e.g., \( t_{31} \) in block \( B_1 \)), except for one, all entries of this tuple are removed. By using this approach, three x-tuple matchings result: \((t_{31}, t_{32})\) block \( B_1 \), \((t_{21}, t_{22})\) block \( B_2 \) and \((t_{22}, t_{32})\) block \( B_3 \).

![Fig. 14. Blocking with alternative key values](image)

VI. CONCLUSION

Since many applications naturally produce uncertain data, probabilistic databases have become a topic of interest in the database community in recent years. In order to combine the data from different probabilistic data sources, an integration process has to be applied. However, an integration of uncertain (esp. probabilistic) source data has not been considered so far and hence is still an unexplored area of research.

In order to obtain concise integration results, duplicate detection is an essential activity. In this paper, we investigate how duplicates can be detected in probabilistic data.

We consider probabilistic data models representing uncertainty on tuple and attribute value level with and without using the x-tuple concept. We introduce methods for attribute value matching and decision models for both types of models. Furthermore, we examine how existing heuristics for search space reduction, namely sorted neighborhood method and blocking, can be adapted to probabilistic data.

In conclusion, this paper gives first insights in the large area of identifying duplicates in probabilistic databases. Individual subareas, e.g., detecting duplicates in complex probabilistic data, have to be investigated in future reflections. Moreover,
for realizing an integration of probabilistic data: schema matching, schema mapping and data fusion have to be considered w.r.t. probabilistic source data in future work. Finally, in this paper we consider duplicate detection as a determined process (two tuples are either duplicates or not). Nevertheless, by using a probabilistic data model for the target schema, any kind of uncertainty arising in the duplicate detection process (e.g., two tuples are duplicates with only a less confidence) can be directly modeled in the resulting data by creating mutually exclusive sets of tuples. For that purpose, the used probabilistic data model must be able to represent dependencies between multiple sets of tuples. For example, in the ULDB model dependencies between two or more x-tuple sets can be realized by the concept of lineage.

REFERENCES