On-the-Fly Entity-Aware Query Processing in the Presence of Linkage

Ekaterini Ioannou
L3S Research Center
Hannover, Germany
ioannou@L3S.de

Wolfgang Nejdl
L3S Research Center
Hannover, Germany
nejdl@L3S.de

Claudia Niederée
L3S Research Center
Hannover, Germany
niederee@L3S.de

Yannis Velegrakis
University of Trento
Trento, Italy
velgias@disi.unitn.eu

ABSTRACT

Entity linkage is central to almost every data integration and data cleaning scenario. Traditional techniques use some computed similarity among data structure to perform merges and then answer queries on the merged data. We describe a novel framework for entity linkage with uncertainty. Instead of using the linkage information to merge structures a-priori, possible linkages are stored alongside the data with their belief value. A new probabilistic query answering technique is used to take the probabilistic linkage into consideration. The framework introduces a series of novelties: (i) it performs merges at run time based not only on existing linkages but also on the given query; (ii) it allows results that may contain structures not explicitly represented in the data, but generated as a result of a reasoning on the linkages; and (iii) enables an evaluation of the query conditions that spans across linked structures, offering a functionality not currently supported by any traditional probabilistic databases. We formally define the semantics, describe an efficient implementation and report on the findings of our experimental evaluation.

1. INTRODUCTION

Recent developments in data publishing have brought new requirements and new ways through which users interact with data. In recent days, the typical Internet user is no longer the spectator that consumes information made available by data providers. Through social Web applications, such as *MySpace*, *Blogosphere*, *Facebook*, and through services such as *Twitter*, the user is becoming an active data producer. The creation of feeds and web page aggregators through mashups (e.g., *Yahoo! pipes*), have advanced even further the data production capabilities of average users. Such technologies allow users to develop applications that integrate data from heterogeneous sources in ways that better fit their individual needs, and all this even in the absence of high technical skills.

In this highly heterogeneous environment of the modern web, the existence of different artifacts modeling the same real world object, such as a person, a movie, or a geographical location, is highly prevalent. As it has always been the case, any successful integration task that aims to guarantee the quality of the integrated data, needs to identify these pieces of information and merge them into a single artifact that will be retrieved during query answering. Original integration efforts have been based on keys or mappings [27]. The exponential growth of the number and heterogeneity of sources limited the applicability of such solutions.

A more viable solution was to perform the integration first and then perform a data cleansing operation [12] on the integrated data. The cleansing operation is based on the ability to identify structures representing the same real world objects. There has already been a significant amount of research on this challenge. It can be found in the literature under different names, such as entity linkage [22], merge-purge [21], deduplication [30], entity identification, reference reconciliation [14], or entity resolution [33]. The solutions follow various directions such as measuring structural similarities among data structures [17], string similarities [8], inner-relationships [14, 24], clustering [7], and blocking [33]. These techniques have two major limitations. First, they assume that the data is relatively static. Under this assumption, the process of entity linkage is performed offline. Once performed, the structures found to describe the same objects are merged. Query answering is then performed on the merged dataset [17]. When the original data is highly volatile, this process needs to be continuously repeated and it becomes inefficient. The second limitation is that none of these techniques gives 100% certain results. They typically merge structures that are found to have a belief above some specific threshold value, where the adequate selection of this value is not easy and is often done experimentally.

The evolving nature [29] and the heterogeneity of the data have led to the creation of integration systems[23] that have uncertainty built right into the core of their data processing. They are referred to as *probabilistic databases* [10]. They are databases in which every attribute of an entity or a tuple can have a number of alternative values, each with some probability [11, 28]. A probabilistic database is the result of an integration with uncertain alternative tuples [1, 10, 31], or a consequence of probabilistic mappings from the sources to the integration [16]. Due to these probabilities, query answers are unavoidably probabilistic. Existing techniques are either based on a complete independence assumption among the data records, or support only very specific correlations, e.g., the mutually exclusive property [3]. In dataspaces [16] and data integration scenarios where complementary information may be coming from different sources and needs to be merged, this is not always a satisfying assumption.

In this work we propose a novel technique for query answering under data uncertainty that combines the ideas of entity linkage and probabilistic databases, and extends them to achieve better query answering results under linkage uncertainty. More specifically, first we accept data with uncertainty at the attribute level. Second, we

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were presented at The 36th International Conference on Very Large Data Bases, September 13-17, 2010, Singapore.
Proceedings of the VLDB Endowment, Vol. 3, No. 1
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

obtain the alternative matching suggestions and their beliefs as returned by existing entity linkage techniques, but instead of using them in taking matching decisions off-line, we store this information alongside the data. This leads to a new form of a probabilistic database that apart from uncertainty at the attribute (or tuple) level, contains uncertainty at the merging pairs. The stored uncertainty is used at query time to perform merging of data structures on the fly. This technique has several benefits. First it avoids pitfalls that may result from the one-time a-priori merging decisions, as happens with the traditional entity linkage techniques. Furthermore, it can easier support highly volatile data. The reason is that since no merging decisions have taken place, the only updates required are on the linkages related to the new or modified data. In addition, our technique produces additional valid query answering results compared to those of entity linkage and probabilistic databases, and cannot be simulated following any of those techniques. It is an interesting feature, that reasoning about the entity linkages is done on the fly, which means that some query results may not be explicitly represented in the database but might be a product of the reasoning which is based on the data as well as on the query conditions.

The remainder of this paper is structured as follows. Section 2 provides a motivating example that illustrates the need for query answering based on heterogeneous data with correlations, and with uncertainty on the attributes and on the correlations. Section 3 presents an appropriate data model for the modern Web reality based on entities, instead of relational or semi-structured data. The query answering technique is formally presented and studied in Section 4. Section 5 illustrates its effectiveness and practicality through a series of experimental evaluation steps, and reports on the findings. Section 6 presents conclusions and future work.

2. MOTIVATING EXAMPLE

Consider the entities of an integration system, a fraction of which is illustrated in Figure 1. The entities have been identified and collected from a number of distributed sources. Since the sources may contain outdated or inconsistent data, the attributes of each retrieved entity are coming with some degree of belief. This is modeled in the figure by the numeric values on the attributes. To perform a successful integration, those entities representing the same real world object need to be identified and merged under one single data structure. Unfortunately, the data heterogeneity has led entity identification (i.e., entity linkage) techniques to return results that are also with some degree of belief (confidence). This is indicated in Figure 1 by the numeric values on the dotted lines between the entities. For instance, it is believed with confidence 0.9 that entities e_1 and e_2 represent the same real world object, since despite the same title, it may be the case that one represents the movie Harry Potter and the other the DVD. (The semantics of these numbers will be discussed at a later point.)

Consider now a user looking for "fantasy" stories by "J. K. Rowling". Clearly the only two entities that satisfy both conditions are e_1 and e_3. A linkage technique with a threshold of 0.7, would have decided to merge entities e_1 with e_2, and entity e_4 with e_5. As a result, the answer to the above query would have contained two entities [17]: One would have been the result of the merging of e_1 and e_2, consisting of the union of their respective attributes, and the other would have been the entity e_3. Consider now an entity linkage analysis suggesting the merging of entities e_1, e_2 and e_3. A probabilistic database approach would create one entity that contains the union of all the three [33], with the uncertainty numbers of their attributes updated to reflect the uncertainty of the linkages as returned by the entity linkage algorithm. Similarly, it can merge e_4 and e_5. Then, the user query can be executed in this probabilistic

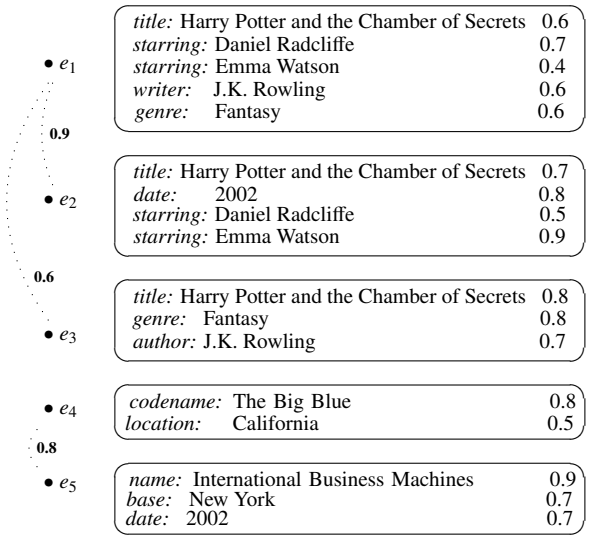

Figure 1: Data source with probabilistic linkage & attributes.

instance [3]. The answer to the query would have contained only one entity, the one derived from the merging of the e_1, e_2 and e_3, and this entity would have been in the result with some derived belief value.

We claim that none of the above two approaches is fully desired. The linkage technique does not take into consideration the fact that there is an even small chance that the entity e_3 is actually modeling the same real world object as the e_1 and should be merged with it as well. The choice of the right threshold is a critical decision. The probabilistic approach, on the other hand, fails to take into consideration that there is a chance that entities e_1 and e_2 and e_3 may be modeling different real world objects and as such, they should not have been merged. If that was the case, then the query on "fantasy" stories by "J. K. Rowling" should have returned two entities in its answer set instead of one. Our position is that in the presence of the uncertainty on the linkage results, leaning towards one solution or the other, results in a loss of information. We believe that a complete answer to the query should take into consideration all the different cases that may exist. In particular, a complete answer should contain three entities, namely, e_1, e_3, and e_{13} which is the merging of entities e_1 and e_3. Each of these entities should of course be in the answer set of the query with some degree of belief, based on the belief of the linkages and the belief of the attributes writer and genre.

One could have taken the above argument even further and claim that the answer set should also contain entities e_{12} and e_{123}, created by the merging of entity e_2 with e_1, or with e_1 and e_3, respectively. We advocate that generating new entities for the result set by including in the merging the attributes of e_2 is redundant since it will simply overwhelm the user with entities that have additional attributes like date=2002 that the user did not ask about in the query. Our position is that no merging should take place unless it is justified by the user query, the linkages and the entity attributes.

The fact that the mergings to be considered in every situation depend on the given query, means that a materialization of the different mergings in advance would have to consider a lot of different cases and, thus, is becoming impractical.

3. ENTITIES WITH LINKAGES

For a data model, we adopt a flexible, entity-based and probabilistic approach. It has the ability to handle highly heterogeneous data, and after its introduction in the context of dataspaces [15]

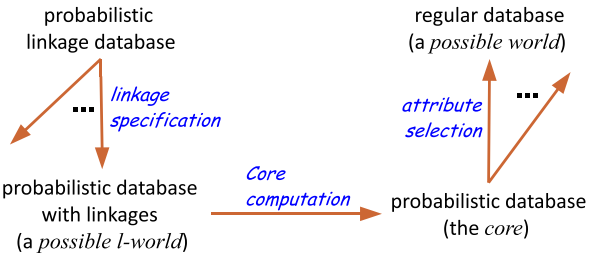

Figure 2: The different kinds of probabilistic databases.

started being used in applications. Its popularity is also based on its similarity to the human way of thinking, which unlocks the potential of developing integration applications for the modern web. A model based on similar ideas can also be found in the literature as *concept model* [9]. Furthermore, the plethora of existing data structures, makes data hard to describe to regular users. This, in combination with the fact that the users often have vague ideas of what they are actually looking for, preferring a more exploratory nature of interaction with the integration systems, leads unavoidably into a simple boolean query language with probabilistic answers.

The fundamental component of our data model is the *entity*, a design artifact used to represent a real world object. An entity is a data structure consisting of a unique identifier and a set of attributes describing its characteristics. Each attribute has a name and a value. The value can be atomic, for example, a string or an integer, but it can also be an identifier of another entity. More formally, assuming the existence of an infinite set of entity identifiers $\mathcal{O}$, an infinite set of names $\mathcal{N}$ and an infinite set of atomic values $\mathcal{V}$, an *attribute* is a pair $\langle n,v\rangle$, with $n\in\mathcal{N}$ and $v\in\mathcal{V}\cup\mathcal{O}$. Let $\mathcal{A}=\mathcal{N}\times\{\mathcal{V}\cup\mathcal{O}\}$ represent the infinite set of all the possible attributes.

Definition 1. *An* entity *e is a tuple* $\langle id, A\rangle$ *where* $id\in\mathcal{O}$ *is the* entity identifier *and* $A\subseteq\mathcal{A}$*, finite, and referred to as the set of* entity attributes. ∎

The ability to use an entity identifier as an attribute value allows the support of relationships among entities. Since each entity is distinguished by its unique identifier, for the rest of the document, the terms entity and entity identifier will be considered equivalent.

A *database* is a set of entities, with each entity partially modeling some part of a real world object. When two entities model parts of the same object, they are said to be *linked*.

Definition 2. *A* linkage database *is a tuple* $\langle\mathcal{E},\mathcal{L}\rangle$*, where* $\mathcal{E}$ *is a finite set of entities and* $\mathcal{L}$ *is a linkage assignment on* $\mathcal{E}$*. A* linkage assignment *over a set E is a binary relation* $L\subseteq E\times E$ *that is commutative, transitive, symmetric, and reflexive. Two entities* $e_1,e_2\in\mathcal{E}$ *of a database* $\langle\mathcal{E},\mathcal{L}\rangle$ *are said to be* linked*, denoted as* l_{e_1,e_2}*, if* $(e_1,e_2)\in\mathcal{L}$*. A maximal group of pairwise linked entities forms a* factor. ∎

A linkage assignment can be equivalently expressed either through explicit enumeration of the binary relationships or through a set of groups of entities, with each group representing a factor. For instance, given entities e_1, e_2, …, e_5, the set $\{\{e_1,e_2,e_3\},\{e_4,e_5\}\}$ describes a linkage assignment with two factors. The first factor consists of entities e_1, e_2, and e_3, and the second of the e_4, and e_5. The alternative representation is through the set of linkages that consists of the set of all pairwise relationships in each factor.

Since two or more linked entities model parts of the same real world object, they can be replaced by a third representing the information of both. In a linkage database this action needs to be followed by the respective update of every reference to these entities found in linkages or attributes, to the newly created entity.

Definition 3. *A* merge *of a set of entities* e_1, e_2, …, e_n, *, denoted as merge*$(e_1, e_2, \ldots, e_n)$*, is a new entity* $e_{new}=\langle id, A\rangle$ *where id is a new identifier and* $A=\cup_{i=1}^{n}A_i$*, with* A_i *representing the attributes of the entity* e_i*.*

The result of a merge *m of entities* $e_1,e_2,\ldots,e_n$ *into* e_{new} *in a linkage database D, is a new linkage database D′ constructed by (i) eliminating from D all entities* $e_1,e_2,\ldots,e_n$*, introducing entity* e_{new}*, (ii) for* $k,m=1..n$ *eliminating all linkages of the form* (e_k,e_m)*, (iii) replacing any linkage of the form* (e_k,e) *(respectively* (e,e_k)*) with* (e_{new},e) *(respectively* (e,e_{new})*), and (iv) replacing every entity attribute of the form* $\langle na,e_k\rangle$ *with* $\langle na,e_{new}\rangle$*. This relationship between D and D′ is denoted as* $D\xrightarrow{m}D'$*.*

The core *of a linkage database D is a database* D_c *such that there is a sequence* m_1, m_2, …, m_m *of merge operators such that* $D\xrightarrow{m_1}D_1\xrightarrow{m_2}\ldots\xrightarrow{m_m}D_c$ *and no other merge is possible on* D_c*.* ∎

By definition, the set of linkages in a core is always empty. It can be shown that the core of a linkage database is always unique but different linkage databases may have the same core.

To capture the uncertainty that may exist on the data, we adopt and extend the idea of the probabilistic databases. First, we associate to each entity attribute a value between 0 and 1. In the absence of linkages, this results into a traditional *probabilistic database* [11]. A probabilistic database D represents a set of *possible worlds*, each being a database in which only a fraction of the attributes in D are present in each entity. An attribute probability indicates the likelihood that an attribute[1] is present in a randomly selected possible world. In the presence of linkages among the entities, we have a *probabilistic database with linkage relationships*. A probabilistic database with linkage relationships is also representing a set of possible databases, i.e., a set of possible worlds. This set is the set of possible worlds of its core.

We push the idea of the probabilistic databases beyond the traditional definition, by introducing uncertainty also on the linkages among the entities. This uncertainty exists naturally from the entity identification or deduplication techniques. The result is a new form of database, referred to as a *probabilistic linkage database*, which is a linkage database with probabilities associated on its linkage relationships and entity attributes.

Definition 4. *A* ***probabilistic linkage database*** *is a tuple* $\langle\mathcal{E},\mathcal{L},p^a,p^l\rangle$*, where* $\mathcal{E}$ *is a set of entities,* $\mathcal{L}$ *is a linkage assignment on* $\mathcal{E}$*, and* p^a*,* p^l *are attribute and linkage probability assignment functions respectively. In particular,* $p^l|_{\mathcal{L}}\mapsto[0,1]$ *and* $p^a|_B\mapsto[0,1]$ *with* $B=\{a\mid\exists\langle id,A\rangle\in\mathcal{E}\wedge a\in A\}$*.* ∎

Due to the probabilities on the linkages, a probabilistic linkage database models a number of different probabilistic databases with linkage relationships. Each such database is generated from the probabilistic linkage database by selecting a fraction of its linkages. We refer to these probabilistic databases with linkage assignments as *possible linkage worlds*, or *possible l-worlds* for short. The set of all possible *l*-worlds of a probabilistic linkage database D is denoted by $plw(D)$.

A possible *l*-world of a probabilistic linkage database is specified by a linkage specification which determines what linkage relationships should be kept and what should be dropped. Not all the specifications are semantically meaningful. For instance, a specification that accepts a linkage between entities e_1 and e_2, and between e_2 and e_3 but not one between e_1 and e_3 is not semantically meaningful since the latter contradicts the first two from which it can be inferred that e_1 and e_3 are linked due to the transitivity property.

Definition 5. *Given a probabilistic linkage database* $D=\langle\mathcal{E},\mathcal{L},p^a,p^l\rangle$*, a* linkage specification *is a linkage assignment* $\mathcal{L}^{sp}\subseteq\mathcal{L}$ *such*

[1]By abuse of terminology, the term *attribute* refers to the pair attribute name- attribute value.

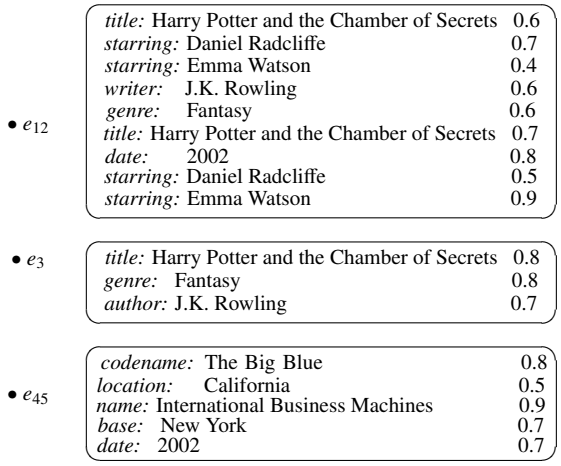

Figure 3: The core of the *l*-world of the database in Figure 1 generated by the linkage specification $\{l_{e_1,e_2}, l_{e_4,e_5}\}$.

that $\forall x \in \mathcal{L}^{sp}$: $p^l(x) \neq 0$. *The* boolean expression *of a linkage specification is the expression* $\bigwedge_{1,n} c_k$, *where*

$$c_k = \begin{cases} x \neq y & \text{if } (x,y) \in \mathcal{L} \wedge (x,y) \notin \mathcal{L}^{sp} \\ x = y & \text{if } (x,y) \in \mathcal{L}^{sp} \end{cases}$$

A linkage specification is invalid *if its boolean expression is always false.* ∎

As an example, consider a database with a linkage assignment $\mathcal{L}=\{l_{e_1,e_2}, l_{e_2,e_3}, l_{e_1,e_3}\}$, and a linkage specification $\mathcal{L}^{sp}=\{l_{e_1,e_2}, l_{e_2,e_3}\}$. Independently of what the probabilities of the linkages are, the boolean expression of $\mathcal{L}^{sp}$ is $e_1 = e_2 \wedge e_2 = e_3 \wedge e_1 \neq e_3$ which is always false, thus the linkage specification is invalid. In our work we consider *l*-worlds constructed by valid linkage specifications only. By definition, given a probabilistic linkage database, and a possible *l*-world of it, there is only one linkage specification defining this possible *l*-world.

We will use the symbol f_i to refer to the i-th factor, and $\mathcal{L}^{sp}_{f_i}$ to denote all the possible linkage specifications between its entities. The k-th assignment in $\mathcal{L}^{sp}_{f_i}$ is denoted by $\mathcal{L}^{sp}_{f_i}(k)$. Since the factors are independent of each other, the probability of a possible *l*-world W can be computed by the product of the probabilities of the factor assignments:

$$Pr(W) = \prod_{i=1}^{n} Pr(\mathcal{L}^{sp}_{f_i}(.)) \quad (1)$$

Example 1. *Consider the probabilistic linkage database of Figure 1 and the linkage specification* $\{l_{e_1,e_2}, l_{e_4,e_5}\}$. *The specification generates a possible l-world that is exactly as the database illustrated in Figure 1, but without the probabilities on the dotted lines and without the dotted lines between entities. Its core will be the one illustrated in Figure 3. Entity* e_{12} *is the result of the merge of* e_1 *and* e_2, *while entity* e_{45} *is the result of the merge of entities* e_4 *and* e_5. *Note that our model allows duplication on the attributes, thus, the fact that the same attribute name/value pair appears twice in an entity is not a problem. Elimination of this kind of duplication and consideration of dependencies among the attributes can be handled at a later stage.* ∎

Figure 2 provides a graphical explanation of the relationships among the different types of databases defined here.

It is important to note here that, in general, traditional entity linkage techniques measure beliefs. Some recent works explain how to turn these beliefs into probabilities [3, 16, 17]. We consider this task outside the focus of the current work. We assume that this information has been computed by some data analysis tools [5] or some other form of linkage discovery algorithms [16], and has been provided to our framework as input. Another important note is regarding the meaning of a probabilistic linkage between two entities. A linkage represents the belief that the two entities are linked, independently of any other third entity. It is not a global belief. This means that through different linkage paths, different linkage beliefs may be computed. For instance, consider the simple example of three entities e_1, e_2 and e_3, with the following linkages between them: l_{e_1,e_2}=0.3, l_{e_2,e_3}=0.5 and l_{e_1,e_3}=0.8. Through transitivity, from the linkages l_{e_1,e_2} and l_{e_2,e_3}, it can be inferred a belief that entity e_1 and e_3 are linked with probability 0.3×0.5=0.15 which is different from the 0.8 direct linkage l_{e_1,e_3}. The way all these different probabilities are combined together to form the global belief of linkage between e_1 and e_3 is up to the query mechanism. It can be, for instance, the maximum value, their sum, or something else. In our system, this situation is taken care through the probability computation of the *factor*, that will be presented later.

For a query language we have adopted a flexible formalism that covers the needs of the emerging case of concept databases [9]. In particular, a query is a conjunction of attribute name-value pairs in which the user describes the characteristics, i.e., attributes, that the retrieved entities are expected to satisfy. An answer to a query is a set of entities. Consider a probabilistic linkage database D, and a query Q:$a_1 \wedge \ldots \wedge a_n$, with each a_i being an expression of the form $name_i$=$value_i$. An entity e is in the answer set of Q if there is a possible world W of a probabilistic database D_p, such that D_p is the core of a possible *l*-world $W^{\mathcal{L}}$ of the database D, entity e is in W and it contains an attribute $name_i$ with value $value_i$ for every i=1..n. In other words, the answer to a query is the union of the answers over all the possible worlds of all the possible *l*-worlds. Each entity in the answer set of a query is accompanied with a probability, which represents the belief we have that this entity will be selected among all the possible worlds of all the possible *l*-worlds of the probabilistic linkage database. This probability is computed based on the attribute and linkage probabilities.

4. EFFICIENT QUERY EVALUATION

One of the aspects of the proposed work that needs to be investigated is the evaluation of a query Q over a probabilistic linkage database $\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$. The semantics of query answering suggest an evaluation strategy that consists of the computation of all possible *l*-worlds, for each such *l*-world the computation of its possible worlds, and then an evaluation of the query Q over each such world. For each entity in the answer set, the number of its appearances in the possible worlds can be computed to determine its probability. It is clear that such an evaluation is prohibitively expensive both in terms of space and time.

Instead of the brute-force evaluation, we propose here an alternative evaluation strategy that is based on a novel technique that avoids high computational cost and requires no materialization of worlds. The general high level idea, summarized in Algorithm 1, may seem to be similar to the evaluation of queries in probabilistic databases [31], but the existence of linkages makes the problem fundamentally different. Existing models for *correlated tuples in probabilistic databases* [31], for instance, cannot handle this situation. Despite the correlation they support among tuples, they still consider the tuples independent structures and manage them as such. In our case the linked entities are merged into one.

A distinguishing feature of our approach is that it restricts the computation to only those possible *l*-worlds and their corresponding linkage specifications that are meaningful for the query at hand.

Algorithm 1: Entity-Aware Query Evaluation

Input: Query Q
Output: Set R of Entities satisfying query conditions

```
LA ← findRequiredLinkageAssignments(Q);
PLW ← ∅;
R ← ∅;
foreach la ∈ LA do
    W ← findPossibleLWorlds(la);
    foreach w ∈ W do
        w.prob ← calculateLWorldProbability(la);
        PLW ← PLW ∪ {w};
    foreach plw ∈ PLW do
        E ← evaluateQuery(plw, Q);
        foreach e ∈ E do
            e.prob ← combineProb(e.prob, plw.prob);
            R ← R ∪ {e};
```

As a consequence, it is not simply the entities in the answer set that depend on the query, but also their structure, i.e., the attributes they contain: adding an extra condition to a query may trigger the consideration of additional linkage specifications, which —in turn— may result in additional attributes for the entities in the answer set. This feature has two major benefits. First, it makes the whole process computationally cheaper, since only the required merges take place. Second, it avoids overwhelming the user with answers containing long lists of attributes originating from different possibly linked entities that are outside the users' interests.

Our approach consists of the following steps. First, we build an index on all the factors (Sec. 4.1). Since there is a one-to-one correspondence between possible l-worlds and linkage specifications, and between linkage specifications and entity merges, we start by finding the entity merges required in order to generate an answer to the query at hand (Sec. 4.2). From the merges we find the linkage assignments that need to be considered, and from these assignments the possible l-worlds. Then the probability of each possible l-world is computed (Sec. 4.3). Finally, the possible worlds of each l-world are generated alongside their own probability, which is combined with the probability of the respective l-world and then included in the answer set of the query (Sec. 4.4).

4.1 Representing & Indexing Factors

A commonly used approach [4, 11, 28, 31] in answering queries over probabilistic data is to partition the data into a series of disjoint/independent groups. These groups can be found in the literature under names such as *factors* [31] or *components* [4]. The possible combinations of these groups generate all the possible worlds.

This idea is not directly applicable to our case. The transitive property of linkage may generate additional correlation, i.e., dependencies, that are equally important for the correct identification of the possible worlds.

We do, however, follow a similar idea to the one of managing uncertain data with correlations [31], and as a first step, we divide the set of entities into sets of connected components, i.e., factors (Def. 2). To compute all possible l-worlds of a database, we need to consider all the possible valid linkage specifications. This number can easily get large to make the computation intractable. Based on the fact that no linkage exists between entities in different factors, we can improve the situation by considering each factor independently.

Each possible l-world is based on some linkage specification within each factor. Thus, the set of possible l-worlds can be derived by combining the alternative linkage specifications withing each factor. In other words:

$$plw(\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle) \quad = \quad \mathcal{L}^{sp}_{\mathcal{f}_1} \times \mathcal{L}^{sp}_{\mathcal{f}_2} \times \ldots \times \mathcal{L}^{sp}_{\mathcal{f}_n}$$

The probability of a generated l-world is computed using Formula 1, and is based on the assignments of its linkage specifications. The probability of assignment $\mathcal{L}^{sp}_{\mathcal{f}_i}(k)$ is based on the probabilities of the linkages it uses. We compute it by considering only the linkages that can not be derived, and thus use set $M \subseteq \mathcal{L}^{sp}_{\mathcal{f}_i}$ that does not include more than once the same information. This probability is computed as $\prod_{l_i \in M} (p^l(l_i)) \cdot \prod_{l_i \in (\mathcal{L}^{sp} - M)} (1 - p^l(l_i))$. In this work, we selected to remove linkages that had a lower probability than their corresponding derived one. However, other options can also be incorporated.

Example 2. *In the database of Figure 1, the set of entity linkages is* $\mathcal{L} = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_4,e_5}\}$ *in which two independent factors can be identified:* $\mathcal{f}_1 = \{e_1,e_2,e_3\}$ *and* $\mathcal{f}_2 = \{e_4,e_5\}$. *The first contains entities* e_1, e_2, *and* e_3 *with linkages* $\mathcal{L}^{sp}=\{l_{e_1,e_2}, l_{e_1,e_3}\}$, *and the second contains* e_4 *and* e_5 *with linkages* $\mathcal{L}^{sp}=\{l_{e_4,e_5}\}$. *The sets of possible linkage specifications with the respective probabilities of the l-world they specify are:*

Factor $\mathcal{f}_1 = \{e_1,e_2,e_3\}$			*Factor* $\mathcal{f}_2 = \{e_4,e_5\}$	
$\mathcal{L}^{sp}_{\mathcal{f}_1}(1)=\{l_{e_1,e_2},l_{e_1,e_3}\}$	*0.9×0.6=0.54*	×	$\mathcal{L}^{sp}_{\mathcal{f}_2}(1)=\{l_{e_4,e_5}\}$	*0.8*
$\mathcal{L}^{sp}_{\mathcal{f}_1}(2)=\{l_{e_1,e_2}\}$	*0.9×(1-0.6)=0.36*		$\mathcal{L}^{sp}_{\mathcal{f}_2}(2)=\{\}$	*(1-0.8)=0.2*
$\mathcal{L}^{sp}_{\mathcal{f}_1}(3)=\{l_{e_1,e_3}\}$	*0.6×(1-0.9)=0.06*			
$\mathcal{L}^{sp}_{\mathcal{f}_1}(4)=\{\}$	*(1-0.9)×(1-0.6)=0.04*			

Considering all the possible combination of the above individual linkage specifications of factors, the linkage specifications of the whole database can be constructed. Each such specification, specifies a possible l-world. The following table provides these l-worlds (through the linkages that each one considers) alongside the respective entity merges that need to take place in the computation of the core of the l-world. The meaning of the notation $e \equiv e'$ *is that in the core computation of the respective l-world the merge of entities* e *and* e' *needs to take place.*

Possible l-world	*Required Merges*	*Probability*
$I_1 = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_4,e_5}\}$	$e_1 \equiv e_2 \equiv e_3$, $e_4 \equiv e_5$	$0.54 \times 0.8 = 0.432$
$I_2 = \{l_{e_1,e_2}, l_{e_1,e_3}\}$	$e_1 \equiv e_2 \equiv e_3$, e_4, e_5	$0.54 \times 0.2 = 0.108$
$I_3 = \{l_{e_1,e_2}, l_{e_4,e_5}\}$	$e_1 \equiv e_2$, e_3, $e_4 \equiv e_5$	$0.36 \times 0.8 = 0.288$
$I_4 = \{l_{e_1,e_2}\}$	$e_1 \equiv e_2$, e_3, e_4, e_5	$0.36 \times 0.2 = 0.072$
$I_5 = \{l_{e_1,e_3}, l_{e_4,e_5}\}$	$e_1 \equiv e_3$, e_2, $e_4 \equiv e_5$	$0.06 \times 0.8 = 0.048$
$I_6 = \{l_{e_1,e_3}\}$	e_2, $e_1 \equiv e_3$, e_4, e_5	$0.06 \times 0.2 = 0.012$
$I_7 = \{l_{e_4,e_5}\}$	e_1, e_2, e_3, $e_4 \equiv e_5$	$0.04 \times 0.8 = 0.032$
$I_8 = \{\}$	e_1, e_2, e_3, e_4, e_5	$0.04 \times 0.2 = 0.008$

The sum of the probabilities of the possible l-worlds in the above table is 1. In certain cases, the sum could have been less. This is because of the fact that certain linkage specifications are not valid and are not considered. This would have been the case in the specific example, for instance, if there was also a linkage between entities e_2 *and* e_3. ∎

To avoid recomputing the factors every time, we create an index structure that is dynamically maintained. The index is based on the idea of equivalence classes. Actually, each factor is an equivalence class. When the data is modified and new linkages are introduced or old are eliminated, changes occur on the equivalence class memberships, and thus on the factors.

4.2 Deciding the Entity Merges

Clearly, not all the possible l-worlds need to be created every time a new query needs to be answered. If the core of a possible l-world contains no entity that satisfies all the attributes requested in

Algorithm 2: Generate Entity Merges

Input: Query $Q := \langle a_1, a_2, \ldots, a_k \rangle$, Database $\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$
Output: Entity Merges M

```
foreach a_i in Q do
    E_i ← {e | e=⟨id, A⟩ ∧ e∈ℰ ∧ a_i∈A};
N ← {(e_1, ... e_n) | ∀i=1..n: e_i∈E_i ∧ ∀i=2..n: factor(e_{i-1})=factor(e_i) };
M ← {eliminateDuplicates(m) | m ∈ N} ;
```

the query, then it is certain that every possible world of the core will return no answer to the query. To avoid these l-worlds, we exploit the list of factors that have been precomputed and indexed. From all the possible factors, only those that for every attribute mentioned in the query, contain at least one entity satisfying that attribute, are considered.

The above step already provides a considerable reduction to the number of linkages and entities that need to be processed. However, merging all the entities in each of the selected factors may result into entities with a large number of attributes. We exploit the linkage specifications to push the optimization even further by considering only the linkage specifications of each factor that are between entities satisfying at least one attribute from those in the query. Furthermore, it is required that the union of the attributes of the factor entities involved in the linkages of the linkage specification to be a superset of the set of attributes in the query. In practice the above selections and the merges are actually computed in one step.

Algorithm 2 provides the steps we follow. For each query attribute we create a set E_i with all the entities satisfying the specific attribute. Then we create the cartesian product of these sets, with the extra requirement that the entities should belong to the same factor. Since an entity may belong to more than one E_i set, we involve a duplicate elimination step at the end.

Example 3. *Consider the query Q: starring="Emma Watson" ∧ date=2002 on our usual probabilistic linkage database example of Figure 1. Only entities e_1 and e_2, both belonging to factor f_1, satisfy the first attribute starring="Emma Watson", thus, the list E_1={f_1-e_1, f_1-e_2} is constructed. Similarly, the list E_2={f_1-e_2, f_2-e_5} is also constructed for the attribute date=2002 of the query. The cartesian product of these two lists, with the additional condition of agreement on the factors gives the pairs: $\langle f_1-e_1, f_1-e_2 \rangle$ and $\langle f_1-e_2, f_1-e_2 \rangle$ which becomes $\langle f_1-e_2 \rangle$. This suggest one merging of e_1 and e_2 and one that considers e_2 with no merging. Both make sense since we cannot distinguish between the attribute starring="Emma Watson" of e_1 and that of e_2. However, notice that although e_3 also belongs to factor f_1, it is not considered, since it contains no attribute that has been asked by the query.* ∎

4.3 Computing l-world probabilities

Having decided the merges that need to be performed for satisfying the given query, the next step is to compute the probabilities of the respective l-worlds. Equation 1 under the conditions from a merge that needs to take place, becomes:

$$Pr(I|c_m) = \prod_{i=1}^{m} Pr(\mathcal{L}^{sp}_{f_i} \mid c_m) \quad (2)$$

where c_m are the conditions describing merge m. Recall, however, that a merge may be true in many possible l-worlds. There are two alternatives that one can follow. The first is to compute the probability of the merge as the sum of the probabilities of all l-worlds that satisfy this merge. The second is to compute and consider only the maximum of these probabilities. The latter requires significantly less computation time, since it only needs to identify the l-world with the highest probability. For systems that simply use that probability as a ranking mechanism for the entities before displaying them to the user, this second option is typically sufficient.

aid.	name	value	p
• a_{10}	starring	Daniel Radcliffe	0.7
◇ a_{11}	starring	Emma Watson	0.4
a_{12}	writer	J.K. Rowling	0.6
a_{13}	genre	Fantasy	0.6
• a_{20}	starring	Daniel Radcliffe	0.5
◇ a_{21}	starring	Emma Watson	0.9

Possible Worlds

(1)	(2)	(3)	(4)
a_{10}	a_{20}	a_{10}	a_{20}
a_{11}	a_{11}	a_{21}	a_{21}
a_{12}	a_{12}	a_{12}	a_{12}
a_{13}	a_{13}	a_{13}	a_{13}

Figure 4: Possible worlds for *merge*($e_1 \equiv e_2$) for exclusive attributes.

The algorithm for computing the maximum probability is based on the algorithm for finding shortest paths in graphs. In particular, provided the entity linkages $\mathcal{L}^{sp}_{f_i}$, we generate a weighted undirected graph G as follows: every entity participating in a linkage of l_{e_i,e_j} becomes a node of the graph. Each linkage l_{e_i,e_j} becomes an edge that connects the nodes representing entities e_i and e_j. The weight of such an edge is given by the probability of the respective linkage.

An entity merging *merge*(e_1,e_2,...,e_n) corresponds to a spanning tree that connects all entities e_1,e_2,...,e_n. Computing the merging that maximizes the probability is similar to computing the maximum connected component of the graph that has the highest total probability (i.e., multiplication of the probabilities of its edges). Since the nodes of the graph correspond to the entities of a factor, they are all connected, thus, the maximum connected component will include all the nodes of the graph. To compute it, we rank the edges in decreasing order of their linkage probability. Initially, all the entities (i.e., nodes) are marked as not-visited. The highest ranked edge is first selected and the two nodes it connects are marked as visited. Then as a list of edges is considered the subset of the edges that have one endpoint marked visited and one non-visited. The one with the highest probability is selected and its non-visited endpoint is marked as visited. The same step is repeated until all the nodes in the graph have been marked as visited. The probability of the merge is the multiplication of the probabilities of the edges that have been used in this process, and by construction this probability is the maximum.

4.4 Possible worlds and their probabilities

Each possible world from an l-world essentially represents a different combination over the attributes of the entities participating in a specific l-world. For instance, consider the data in Figure 3, and in particular the attributes involved in *merge*(e_1,e_2). The entity *merge*(e_1,e_2) needs to include all attributes from entities e_1 and e_2, as shown in Figure 4. Two issues need to be taken into consideration. One is the probabilities of the attributes, specifically in the case of duplication, and the other is the dependencies that may exist among them.

The attributes that appear in real world datasets are not always independent. The correlations (i.e., dependencies) between attributes that need to be considered in attribute merge, strongly depend on the nature of the sources and their datasets. Our framework is able to handle such correlations in a uniform manner. The following paragraphs provide more details for generating worlds with two possible collections:

A. Independent Attributes. One option is to assume no correlation and thus no restrictions on which attributes to include in the resulted entity merge. This case results is only one world given by the union of all attributes: merge(e_1, ..., e_n) = ⟨ id', $\cup_{i=1}^{n} e_i.A$ ⟩.

B. Exclusive Attributes. In certain cases, the attributes originating from different entities participating in the entity merge are exclu-

sive. This requires that only one occurrence of such an attribute to be in the entity resulted by the merge. A typical example of such an attribute are the distinct attribute names, e.g., a person can have only one name. Other examples are the attributes with the same name but similar (semantically or syntactically) values, e.g., attributes a_{11} and a_{21} from Figure 4. A simple method is to cluster the exclusive attributes from each entity, i.e., $M = \{\{e_1.\alpha_i, e_1.\alpha_j, \ldots\}\}$. We can then use this set to generate worlds with these correlations: merge$(e_1, \ldots, e_n) = \langle$ id', $A \rangle$, where $A \subseteq (\ M_1 \times M_2 \times \ldots \times M_m\ ) \cup \{\ \alpha \mid \alpha \notin \cup_{i=1}^{m} M_i.\alpha\ \}$.

The overall probability of a possible world depends on the probability of the attributes included or not included in the world. It is computed as the product of probability p^α when attribute α is part of the world and $(1 - p^\alpha)$ when attribute α is not part of it: $\Pr(e'|\text{merge}(e_1, \ldots, e_n)) = \Pr(l\text{-world}) \times \prod_{\alpha \in e'.A} p^\alpha \times \prod_{\alpha \notin e'.A\ \&\ \alpha \in e_i.A} (1\text{-}p^\alpha)$.

Example 4. *Fig. 4 shows the attributes involved in merge(e_1,e_2). The exclusive attributes are given by set M=$\{\{\alpha_{10},\alpha_{20}\},\{\alpha_{11}, \alpha_{21}\}\}$. Fig. 4 shows the four generated possible worlds, and their probability is computed according to above formula.*

5. EXPERIMENTS

We have performed a number of experiments in order to study effectiveness and efficiency of our approach, referred to as **EAQP**, in comparison to two other approaches. The first represents the existing entity linkage techniques, and is referred to as **ELA** in this section. This approach performs off-line merging of the data with linkage probability above some predefined threshold, known as the *entity linkage threshold*. For a fair comparison and following the suggestion of [33], we kept all properties of the matched entities as part of the respective merged entity. The merged entity is thus a union of those attributes pairs, increasing the probability that a query information matches one of the entity attributes. The second methodology with which we compared our approach is the work on probabilistic databases, referred to as **PDBA**. Since PDBA solutions consider probabilities for attributes or tuples only, but not for linkages, a direct comparison with our approach with respect to the effectiveness would not have been fair. Thus, the only comparison we did between EAQP and PDBA was in terms of efficiency. All evaluations are reported on the average of 800 queries, constructed by randomly selecting object attributes.

Cora Dataset. It is a collection of publications and authors from CiteSeer[2], that is typically used to evaluate linkage techniques [3, 14, 32]. It contains 9,774 author descriptions that refer to 2,882 real world objects. We generated entity linkages between authors (i.e., entities) using the probabilistic entity linkage algorithm [22]. Fig. 5 shows the number of linkages for different linkage thresholds. Precision and recall of the generated entity linkages are similar to the ones generated by other algorithms such as [14, 32]. For our approach we did not apply a threshold in order to obtain all the linkages, even those with low probabilities.

Movie Dataset. For evaluating the efficiency we needed a sufficiently large dataset and also linkages coming from different linkage techniques. We generated such a dataset by integrating data describing 13,435 movies coming from two real world systems: 23,182 *IMDb* movies (relational data) and 28,040 *DBpedia* movies (RDF data). We converted both datasets to our data model and stored them in a relational database. For generating the entity linkages we compared the movie titles using two standard string similarity methods [8], *Jaccard* and *Jaro*. Fig. 6 plots the precision-recall graph resulting when using these techniques to link entities, with *Jaccard* being more successful in linking *IMDb* to *DBpedia* movies than *Jaro*. As expected, for both techniques we see the typical dependency between precision and recall. Linkage techniques always have to make a trade-off between the two. In our experiments we investigate how our approach addresses this issue.

[2]http://www.cs.umass.edu/~mccallum/data/cora-refs.tar.gz

Entity Linkages (under threshold t)					
t=0.52	t=0.58	t=0.62	t=0.68	t=0.72	t=0.78
12,440	12,012	10,775	6,394	5,985	4,184

Figure 5: Linkages in the Cora datasets.

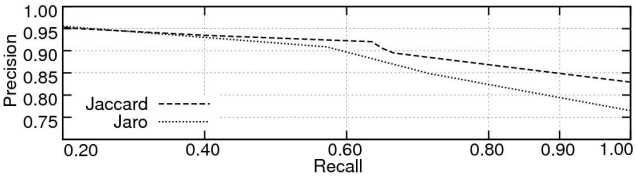

Figure 6: Precision-recall for the movie datasets.

Effectiveness. In our first experiment we examine the result quality of our approach using the Cora dataset. We processed the 800 queries and compared the results returned by EAQP and by ELA. Evaluation of the effectiveness is based on the ground truth of the Cora Dataset. For ELA, queries are evaluated over the already pre-merged entities (based on the respective entity linkage threshold). Selecting a low threshold (t=0.6) will provide linkages with a high recall but low precision, whereas selecting a higher threshold (t=0.8) will provide linkages with significantly higher precision, but with lower recall. So by increasing the threshold only linkages with high probabilities are accepted and the number of linkages, thus, is reduced (see Fig. 5 for the exact numbers). We examine the behavior of EAQP as well as of ELA for increasing linkage threshold values.

Fig. 7 shows the average F-measure (weighted harmonic mean of precision and recall) of the 800 queries for various entity linkage thresholds. As expected, when moving towards higher thresholds, the entity linkage technique accepts less and less linkages. This makes the technique unable to find the entities described by the queries. EAQP exhibits a higher F-measure than ELA for the entire range of considered linkage threshold values. The difference is especially high for linkage threshold between threshold values 0.65 and 0.75, where EAQP is still able to identify many of the searched entities. For instance, for t=0.66 EAQP returned the correct entity for around 10% more queries than ELA. This is because EAQP can find connecting linkages to construct the entity described in the query, even if the linkage probability is below the threshold. ELA had to reject these linkages due to their low threshold.

Fig. 10 shows the numbers of queries that were correctly answered for different linkage thresholds. As shown query processing with our approach returns the correct results to more queries than ELA. In additional experiments we performed, we noticed that the entities returned by EAQP were with higher confidence (i.e., with higher probability) than the entities returned by ELA. For instance, for t=0.6 EAQP returned 421 correct answers whereas ELA returned 238 correct answers. For 91 answers, the entities of EAQP had higher probability than the entities of ELA.

Efficiency. The core of our approach is based on generating factors by grouping entities which are pairwise linked. During query processing we select the factors to construct the entities relevant for the query. The number of entities in the factors influences the execution time of our approach. In this experiment we examined this influence using the Movie dataset. We computed the size of the generated factors as the number of entity linkages contained

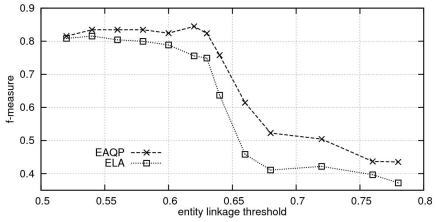

Figure 7: F-measure for EAQP and ELA.

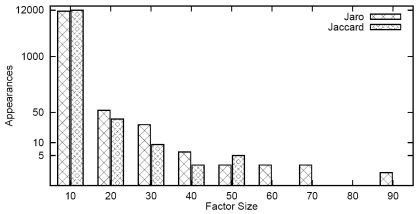

Figure 8: Data statistics for the factors.

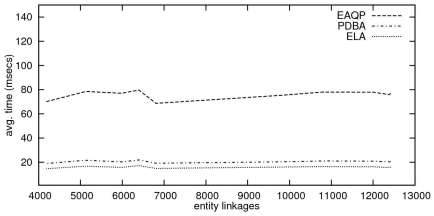

Figure 9: Query answering time.

in the factors. We then constructed the histogram of factor sizes. Fig. 8 shows appearances per factor sizes as generated by both entity linkage techniques, *Jaro* and *Jaccard*. It can be seen that only few factors have a large size, which means less overall processing time.

Our final evaluation was to measure the time required for EAQP, and also to compute the overhead that a system will have for offering this additional functionality. Fig. 9 shows the average time taken to answer queries with EAQP, PDBA, and ELA. We show time over different number of entity linkages in dataset. As expected there is an increase in the time required by our approach, but this is relatively small and it remains under 70 milliseconds. Furthermore, time does not increase as the dataset gets larger. On the contrary, query time remains stable even when the data size doubles. This behavior is justified by the effective grouping of linkages into factors that takes place and allows the algorithm to easily detect and use only a small subset of the linkages during query processing. As expected the additional time required by the PDBA is lower than that of EAQP, since PDBA does not cover the full semantics of our approach, especially not considering linkage probabilities. This clearly leads to a much lower number of possible worlds to be considered, which is reflected in the smaller increase in run time.

We have also studied the behavior of the time required of retrieving the possible l-worlds, and have also measured its improvements over ELA, but we will not ellaborate further on the issue.

6. CONCLUSIONS

We have introduced a novel approach for on-the-fly entity-aware query processing in the presence of linkage information. We have formally defined the semantics of a new kind of query answering, and describe efficient techniques for query processing. The results of our evaluation on real world datasets show the efficiency and effectiveness of our approach. Our current work focuses on extending the model with provenance information, and investigating the implications of conflicting entity information, as this sometimes appears in Web data.

Acknowledgment This work has been partially supported by the EU Projects OKKAM ICT-215032 and Papyrus GA-215032.

References

[1] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. U. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, 2006.

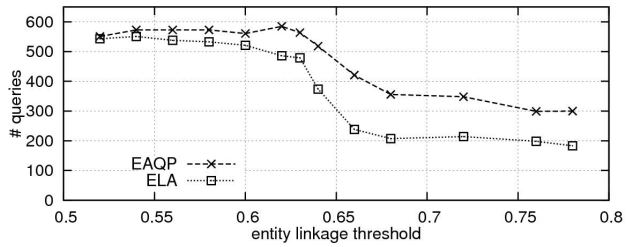

Figure 10: Query success of EAQP and ELA.

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, pages 586–597, 2002.
[3] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE*, 2006.
[4] L. Antova, C. Koch, and D. Olteanu. 10^{10^6} worlds and beyond: Efficient representation and processing of incomplete information. In *ICDE*, 2007.
[5] G. J. Bex, F. Neven, and S. Vansummeren. Inferring xml schema definitions from xml data. In *VLDB*, pages 998–1009, 2007.
[6] I. Bhattacharya and L. Getoor. Deduplication and group detection using links. In *LinkKDD*, 2004.
[7] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD*, pages 11–18, 2004.
[8] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78, 2003.
[9] N. N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *PODS*, pages 1–12, 2009.
[10] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB*, 16(4):523–544, 2007.
[11] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, pages 1–12, 2007.
[12] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.
[13] A. Doan and A. Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005.
[14] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96, 2005.
[15] X. Dong and A. Y. Halevy. Indexing dataspaces. In *SIGMOD*, pages 43–54, 2007.
[16] X. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In *VLDB*, pages 687–698, 2007.
[17] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.
[18] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 2005.
[19] R. V. Guha and R. McCool. Tap: a semantic web platform. *Computer Networks*, 42(5):557–577, 2003.
[20] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, pages 1–9, 2006.
[21] M. A. Hernández and S. J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.*, 1998.
[22] E. Ioannou, C. Niederée, and W. Nejdl. Probabilistic entity linkage for heterogeneous information spaces. In *CAiSE*, pages 556–570, 2008.
[23] E. Ioannou, C. Niederee, and Y. Velegrakis. Enabling Entity-Based Aggregators for Web 2.0 data. pages 1119–1120, 2010.
[24] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Trans. Database Syst.*, 31(2):716–767, 2006.
[25] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD Conference*, pages 802–803, 2006.
[26] A. M. Ouksel and A. P. Sheth. Semantic interoperability in global information systems: A brief introduction to the research area and the special section. *SIGMOD Record*, 28(1):5–12, 1999.
[27] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernandez, and R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.
[28] C. Re and D. Suciu. Managing probabilistic data with mystiq: The can-do, the could-do, and the can't-do. In *SUM*, pages 5–18, 2008.
[29] F. Rizzolo, Y. Velegrakis, J. Mylopoulos, and S. Bykau. Modeling Concept Evolution: A Historical Perspective. volume 5829, pages 331–345, 2009.
[30] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, pages 269–278, 2002.
[31] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, pages 596–605, 2007.
[32] P. Singla and P. Domingos. Multi-relational record linkage. In *KDD Workshop on Multi-Relational Data Mining*, 2004.
[33] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD Conference*, pages 219–232, 2009.

APPENDIX

A. RELATED WORK

Entity Linkage. Most of the existing entity linkage techniques, focus on off-line identification and linkage of the data describing the same real world objects. A complete overview of the existing work in this domain can be found in surveys [13, 18, 17] and workshops/tutorials [25, 26].

The current state-of-the art in this research area are the algorithms that utilize associations between entity representations to identify linkages. To capture the associations found inside a dataset, data is modeled into supportive structures. Ananthakrishna et al. [2] exploit dimensional hierarchies to detect fuzzy duplicates in dimensional tables. Hierarchies are built by following the links between the data from one table to data of other tables. Entities are matched when the information along these generated hierarchies is found similar. Getoor et al. [6, 7] model the metadata as a graph structure, with nodes being information describing the entities and edges the relationships between the entities. Edges are used to cluster the nodes, and the found clusters help to identify the common entities. In [24], the data is also modeled as a graph following a similar methodology as the previous method. This method also generates other possible relationships (modeled as edges in the graph) to represent the candidate matches between entities. Then, techniques from graph theory are used to analyze the relationships in the graph and decide the correct entity matches.

The TAP system [19] uses a process named *Semantic Negotiation* to identify common descriptions (if any) between the different resources. These common descriptions are used to create a unified view of the data. Another well-know algorithm is the *Reference Reconciliation* [14]. Here, the authors begin the computation by identifying possible associations between entities through comparisons of entity descriptions. The information encoded in the found associations is propagated to the rest of the entities in order to enrich their information and improve the quality of final results. Whang et al. [33] introduced a linkage approach focusing on efficiency, which is achieved through partitioning the data into blocks. For each entity they maintain the complete set of matching data, a method that allows them to discover further matches by iterative over their current set of matches.

Probabilistic Data. Few existing data integration proposals focus on dealing with uncertainty that appears in the data through the applied entity linkage algorithms. More specifically, Dong et al. [16] investigate the use of the probabilistic mappings between the attributes of the contributing sources with a mediated schema. Applying this method on the data from Section 2 would have considered the possible mappings between the attribute names as given by contributing sources with a mediated schema S. This means that "title" attribute of e_1, e_2, and e_3 is mapped to a "Title" attribute from S with a probability to show the uncertainty of each mapping. Querying the mediated schema S is then based on these mappings. For example query "title = Harry Potter..." returns e_1, e_2, and e_3. However, it does not really reflect the expected answer, since the expectation is to merge the data of the entities that describe the same objects. In fact, the probabilistic schema mappings described in this approach, can become an input to our approach by representing them as entity linkage information.

The approach in [3] is more similar to ours, since the focus is not on the schema information but on the actual data. The authors assume that the duplicate tuples for each entity are given. In our motivating example (Section 2) this means that linkages do not have probabilities (i.e., we know if they exist or not) and that all tuples describing alternative attributes have the same identifier, e.g.,:

Identifier	Alternative attr.	Probability
id_x	a_{10}	p_1
id_x	a_{20}	p_2

The tuples that represent the alternative attributes are considered as disjoined. This means that only one tuple for each identifier can be part of the final resulted entity. Our proposal does not impose such restrictions/requirements. We explain how entity linkages can contain correlations and provide an appropriate representation and efficient solution for this.

Other related approaches are Dataspaces [20] and Trio [1]. The main focus of these approaches is to create database systems that support uncertainty along with inconsistency and lineage. To some extend these systems also deal with duplicate tuples and uncertain data. Our approach addresses more challenges of heterogeneous data, mainly by considering linkage/matching on the data (not only on schema information), and also correlations between entities.

Another important aspect of our approach is the efficient management of uncertainty in data; a topic that has received a lot of attention recently. Dalvi and Suciu [10] used the notion of possible worlds to introduce query semantics for independent probabilistic data and presented how to efficiently evaluate queries. The approach by Sen et al. [31] moved towards defining and using different correlations, for example that existence of one tuple implies or disallows the existence of another tuple. The methodology identified used in probabilistic databases, for performing efficient query execution is also followed in our approach. We however also extend this methodology to provide its incorporation into a two level processing, i.e., generating possible words inside the generate possible worlds (as explained in Section 3).

B. UNIQUENESS OF THE CORE

We now show that the core of a linkage database D_c is always unique as derived from D through a sequence of merge operators, i.e., $D \overset{m_1}{\rightarrow} D_1 \overset{m_2}{\rightarrow} \ldots \overset{m_k}{\rightarrow} D_c$. The proof is by induction on the results for the size of merge operators with the smallest number.

Hypothesis: Let E denotes the entities in D, i.e., $\{e_1,e_2,\ldots,e_n\}$. We have to perform two merge operators, merge m_1 on the entities in S_a that results in entity e_a, and merge m_2 on the entities in S_b that results in entity e_b.

Induction base: Entity sets S_a and S_b are subsets of E, and by definition the entities in the merge operators are mutually exclusive, so $S_a \cap S_b = \{\}$. The two possible sequences are as follows:

(1) $D \overset{m_1}{\rightarrow} D\text{-}S_a \cup \{e_a\} \overset{m_2}{\rightarrow} D\text{-}S_a \cup \{e_a\}\text{-}S_b \cup \{e_b\} = D\text{-}S_a\text{-}S_b \cup \{e_a,e_b\}$

(1) $D \overset{m_2}{\rightarrow} D\text{-}S_b \cup \{e_b\} \overset{m_1}{\rightarrow} D\text{-}S_b \cup \{e_b\}\text{-}S_a \cup \{e_a\} = D\text{-}S_a\text{-}S_b \cup \{e_a,e_b\}$

Both sequences result in the same linkage database.

Induction step: We now show that if the hypothesis holds, then it also holds for an arbitrary number of merge operations. So, applying a set merge operators M on a linkage database D in any sequence produces the same core linkage database D_c.

Let $\{m_1, m_2, \ldots, m_k\}$ denote an initial merge sequence. We can create a new sequence M' by swapping two nearby merge operators insider M, i.e., $\{\ldots, m_{i+1}, m_i \ldots\}$. Given the induction hypothesis, we know that the resulted D_c will be the same. With iterative swapping of merge operators in the created sequence, we can generate various sequences (and eventually all possible sequences), with each one of this sequences resulting in the same D_c. □

C. PDBA AND ELA ALGORITHMS

For the experimental evaluation we used a JAVA 1.6 implementation of our approach for entity-aware query processing using linkage information (EAQP). We also used a JAVA implementation

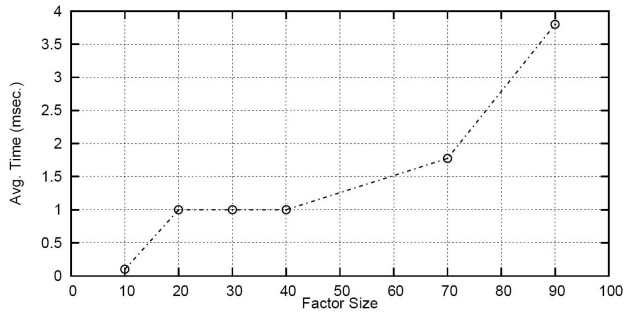

Figure 11: Average time for computing the possible world with the maximum probability over factor sizes.

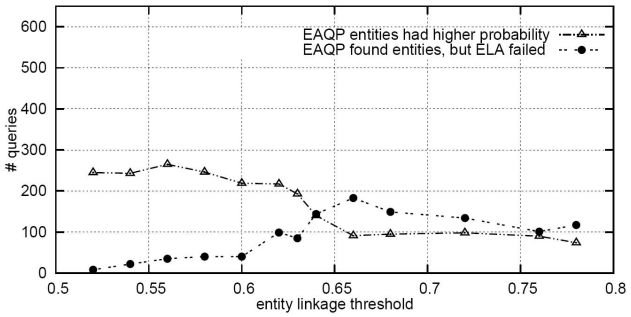

Figure 12: The number of queries, from the ones shown in Figure 10, in which EAQP entities had higher probability, and identified entities whereas the ELA failed.

of two additional methodologies described in the following paragraphs.

Entity Linkage Technique (ELA). This implementation encapsulates the methodology that is currently followed by existing entity linkage techniques. Once a linkage algorithm is applied on the data (e.g., reference reconciliation [14], or entity resolution [33]), a list of the matched data along with the derived matching probability is generated [17, 25]. Then, the entity linkage technique uses a predefined threshold to accept and keep only the linkages those probability is higher than this threshold.

The data of the accepted linkages are consider to exist (i.e., no probabilities), and thus used for creating the finally integrated dataset in which the data found to describe the same real world entity is merged. Matched data is merged using the approach described in [33]. Following this approach, the entities in the dataset are not directly merged and updated, but for each entity we maintain all matched data, and use them when we need to retrieve the entity from the dataset. Therefore, during query processing, an entity will be retrieved even when the query contains data that are not in the final entity representation, since this data is presented in the entity's matched data.

Probabilistic DBs Technique (PDBT). A couple of recent approaches have been introduced for addressing the uncertainty appearing with linkage algorithms [1, 3]. They consider duplicate tuples as alternative representations for the same real world object. These techniques expect that the alternative representations for each entity are known and that only one of them can exist in the final integrated data, i.e., alternative representations are disjoin events. Furthermore, an entity is basically described by the information encapsulated in a single record, and not a set of records.

The existing techniques can not represent and handle the uncertainty on the entity linkage information (Section 1). For being able to perform a comparison of our EAQP approach with PDBT, we converted the problem as it can be represented by the technique introduced in [3]. More specifically, we assume that the attributes of the entity (i.e., characteristics) are given and that each of these attributes have a set of alternative representations. The following tables show how this representation would apply on the data from Figure 1.

id	attr.
e_1	*a*
e_1	*b*
e_1	*c*
e_1	*d*
e_2	...

attr.	name	value	prob.
a	*starring*	Daniel Radcliffe	pr
a	*starring*	Radcliffe, Daniel	pr
b	*starring*	Emma Watson	pr
b	*starring*	Watson, Emma (II)	pr
...			

Although this representation allows us to provide a comparison between the time required for query processing between the EAQP and the PDBT technique, we still need to note the differences in their semantics: (i) PDBT needs to be provided with the exact linkages for then handling possible alternatives for attributes, and (ii) EAQP is able to handle other types of conditions and is not restricted to disjoin events.

D. ADDITIONAL EXPERIMENTS

The following paragraphs present the results of two additional experiments we performed for investigating the efficiency of EAQP and comparing its effectiveness with ELA.

Time to retrieve possible worlds. As we presented in Section 4, our approach separates linkages into factors and query precessing is performed on the related factors. The size of the factors influences the time required for processing queries, so we now investigate the effectiveness for factors of different sizes.

For this experiment, we measured the time needed to identify the possible world with the highest probability in respect to the factor size (Section 4.3). Figure 11 shows the average time required for processing queries over different factor sizes. As expected, for larger factor sizes the algorithm requires more time than for smaller factor sizes, which however still remains below 4 milliseconds. For small factor sizes (i.e., 20-40 entity linkages) that constitute the dominating majority among the factors, the algorithm requires around 1 millisecond.

Improvements over ELA. We further analyzed the results of the evaluation related to effectiveness (Section 5), and identified two situations in which EAQP performs better than ELA. The first is that our approach has less failures, i.e., empty result set as an answer to queries. For instance, for t=0.6 EAQP was able to return the correct answers for the 150 queries in which ELA did not return anything.

The second situation is that there are cases in which the entities returned by EAQP were with higher confidence (i.e., with higher probability) than the entities returned by ELA. As shown in Figure 10, for t=0.6 EAQP returned 421 correct answers whereas ELA returned 238 correct answers. For 91 answers, EAQP had higher probability that ELA. Figure 12 presents the exact numbers for these two situations for various entity linkage thresholds. As shown by the results of the evaluation, EAQP exhibits a higher effectiveness than ELA. There are of course cases in which both approaches return the same answer set. For these cases, we can view the additional processing time of EAQP as a disadvantage in comparison to ELA.