Chapter 9
Embracing Uncertainty in Entity Linking

Ekaterini Ioannou, Wolfgang Nejdl, Claudia Niederée, and Yannis Velegrakis

9.1 Introduction

The modern Web has grown from a publishing place of well-structured data and HTML pages for companies and experienced users into a vivid publishing and data exchange community in which everyone can participate, both as a data consumer and as a data producer. Unavoidably, the data available on the Web became highly heterogeneous, ranging from highly structured and semistructured to highly unstructured user-generated content, reflecting different perspectives and structuring principles. The full potential of such data can only be realized by combining information from multiple sources. For instance, the knowledge that is typically embedded in monolithic applications can be outsourced and, thus, used also in other applications [10]. Numerous systems nowadays are already actively utilizing existing content from various sources such as WordNet or Wikipedia. Some well-known examples of such systems include DBpedia, Freebase, Spock, and DBLife.

A major challenge during combining and querying information from multiple heterogeneous sources is *entity linkage*, i.e., the ability to detect whether two pieces of information correspond to the same real-world object [19, 28]. The task is also important in data cleaning applications [13] and can be found in the literature under different names, such as merge-purge [23], entity identification [29], deduplication [33], data matching [8,15], reference reconciliation [17], or resolution [5]. This topic

E. Ioannou (✉)
Technical University of Crete, University Campus – Kounoupidiana, 73100 Chania, Greece
e-mail: ioannou@softnet.tuc.gr

W. Nejdl · C. Niederée
L3S Research Center, Appelstr. 9a, 30167 Hannover, Germany
e-mail: nejdl@L3S.de; niederee@L3S.de

Y. Velegrakis
University of Trento, Via Sommarive 14, 38123 Trento, Italy
e-mail: velgias@disi.unitn.eu

R. De Virgilio et al. (eds.), *Semantic Search over the Web*,
Data-Centric Systems and Applications, DOI 10.1007/978-3-642-25008-8_9,

has received considerable research attention with many interesting results relying on different methodologies, such as string similarity metrics [8, 9], entity inner-relationships [17, 27], and clustering [7].

Unfortunately, existing approaches for entity linkage assume that data are relatively static. Thus, they typically perform data processing off-line in order to have the results readily available at query time. To achieve this, existing approaches first collect matching evidence, such as similarities between the entity strings or inner-relationships between entities, and based on them, generate information to link the entities, then use predefined thresholds or human intervention to merge the entities. Queries are processed over the resulted merged entities. In modern Web applications, where data may at any time change not only their syntax or structure but also their semantics [35], these techniques are so effective or efficient [19]. This calls for entity linkage techniques that consider and deal with the special characteristics of such data.

This chapter introduces a novel approach for addressing the entity linkage problem for heterogeneous, uncertain, and volatile data. In the following paragraphs, we present the motivation for this work (Sect. 9.1.1), followed by a discussion of the related challenges (Sect. 9.1.2). We then provide an overview of the approach introduced in this chapter (Sect. 9.4), summarize its contributions (Sect. 9.1.4), and finally present the structure of the chapter (Sect. 9.1.5).

9.1.1 Motivation

Consider a system created for monitoring and integrating data from multiple heterogeneous data sources on the Web. The basic data exchange unit of the system is an entity, composed of an identifier and a number of attribute name–value pairs describing the properties of the real-world object the entity represents.

The first part of Fig. 9.1 illustrates three entities existing in the system. The top two entities are referring to the story of Harry Potter and the Chamber of Secrets. The first entity has been extracted through text analysis of Wikipedia articles. Since entity extraction from text is not always accurate, the extracted entity attributes are accompanied with some probabilities reflecting the amount of confidence on the existence of these attributes. In the figure, this confidence is illustrated by the numbers next to the attribute values. The second entity has been extracted from a set of online bookstore databases. A number of these databases contain outdated or inconsistent data; thus, the attributes of the entity are also probabilistic. Finally, the third entity has been extracted by a corpus of news archives. For reasons similar to those of the first entity, its attributes have also some confidence associated with them.

Since the system needs to handle volatile data, we expect continuous appearance of new entities and these entities that need to be integrated with the data already present in the system. The second part of Fig. 9.1 illustrates two additional entities, which the system needs to integrate. Similar to the entities already existing in the

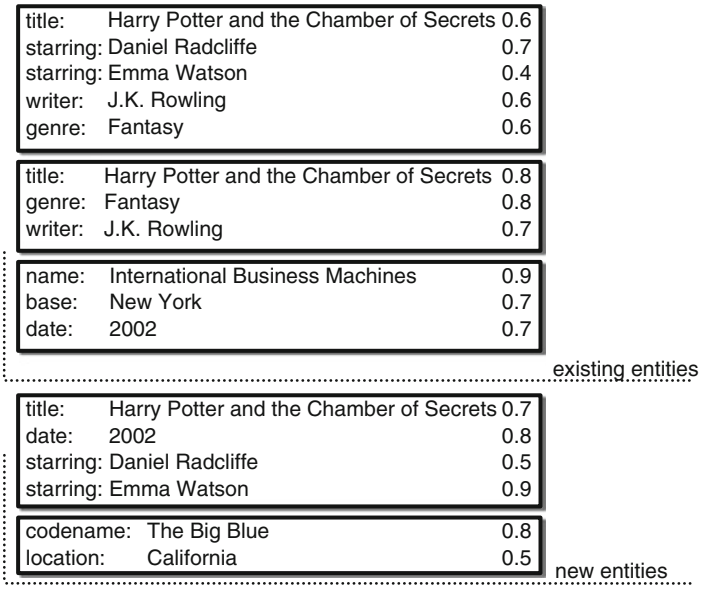

Fig. 9.1 A small fraction of new entities that should be integrated with the entities already existing in the system. Entities are modeled as a set of attributes, i.e., name–value pairs, each with some confidence value that indicates our belief that this attribute describes the specific entity

system, these two entities are also modeled as a set of attribute name–value pairs, each with some confidence value.

A traditional entity linkage methodology [19] would simply use a predefined threshold and accept the merging of these entities when their computed similarity is above this threshold. For the entities of Fig. 9.1, this might mean merging the first two entities. In this situation, a new entity would be created using the data from both entities, which might also involve the removal of some name–value pairs when these are considered as redundant, conflicting, or replicas.

There are two main issues with the traditional entity linkage methodologies. The first is that the system will return no results if asked to return an entity described using some of the attributes that were removed during the merging. The second is that the arrival of a new entity might cause the system to get into a stage that does not accurately reflect reality, since the system is now limited to two options: either decide that the new entity describes the same real-world object as one of those that already exist in the system or that the new entity is not among the already existing entities. This unfortunately ignores the possible options that would arise from reviewing any of the previous merging decisions. As an example, consider a system that has previously performed a merging between entities e_a and e_b, and that it now needs to process the new entity e_c. Merging entity e_c with e_a might provide a better solution than merging it with the previously merging of e_a with e_b. An alternative methodology for considering all possible options would of course be

to maintain the original entities and, at each addition, reexecute an entity linkage technique over the original entities, ignoring the results from previous executions. Unfortunately, this approach has a prohibitively high computational cost, which means that it cannot be applied.

9.1.2 Challenges

To create an effective and efficient solution for the entity linkage problem, we need to consider the characteristics as well as the resulting challenges that appear in the scenarios we are focusing on (Sect. 9.1.1). This can be summarized as follows:

Challenge 1—Volatile Data. Collections created by combining data from various Web applications or extracted data describing resources may constantly change and evolve through interactions with users or external applications. Therefore, the knowledge available to the entity linkage techniques is subject to data reduction, addition, and modification. This implies the need for supporting an incremental computation and adaptation of the linkage information.

Challenge 2—Heterogeneous Information. Effectively addressing the entity linkage problem implies the ability to handle highly heterogeneous data. Dealing with heterogeneity is a task that touches a number of aspects. For instance, the data model for the entities should be able to also capture the possible data heterogeneity. In addition, we need to consider that we are using an entity linkage technique that handles heterogeneous information. The most common methodology to detect entity linkages is based on observing similarities between the attribute values from the entities. However, this assumes that entities describing the same real-world objects would have the same, or at least similar, attribute values. Another methodology relies on identifying and facilitating semantic information, such as relationships between the entities. For example, coauthoring relationship in publications increases the belief that two authors describe the same object. Our solution should therefore allow the combination of results generated by various entity linkage techniques, as a way to capture different linkage methodologies.

Challenge 3—Data Uncertainty. Apart from the uncertainty in the linkage information, data uncertainty also appears for other reasons. One example is data uncertainty that comes directly from the extraction process due to the very low quality that typically accompanies the unstructured data of such applications [21]. Another example is the uncertainty introduced when building structures for processing the data, e.g., social network analysis [1]. These approaches typically affect the quality of data, which is then reflected through probabilities. Unfortunately, incorporating uncertainty in a system may break a number of assumptions that many entity linkage techniques rely upon. Thus, performing entity linkage over uncertain data is a major challenge.

9.1.3 Summary of the Approach

The methodology we follow takes into consideration heterogeneity, uncertainty, and the volatile nature of the data. It is based on maintaining the linkage information among the entities. As an example, let us consider the entities in Fig. 9.1. It is easy to see that the first two entities may represent the same real-world object, for instance, the first entity may represent the actual movie, whereas the second entity is a DVD with the respective movie. Given that we do not have enough evidence to support a definite decision on whether these entities represent the same real-world object or not, we do not perform a merging between them. We compute and store a probabilistic linkage connecting the two entities. The addition of the new entities requires only the computation of the linkages or (in some cases) the recomputation of the probability of existing linkages.

Figure 9.2 illustrates the computed linkages through the interconnecting dotted lines and alongside their probabilities. As depicted in the figure, these entities have three probabilistic linkages, two among the movie entities that are labeled e_1-e_3, and one among the company entities that are labeled as e_4 and e_5. Once we reach a final decision that two or more entities are linked, we can replace them by an equivalent entity consisting of the union of their attributes.

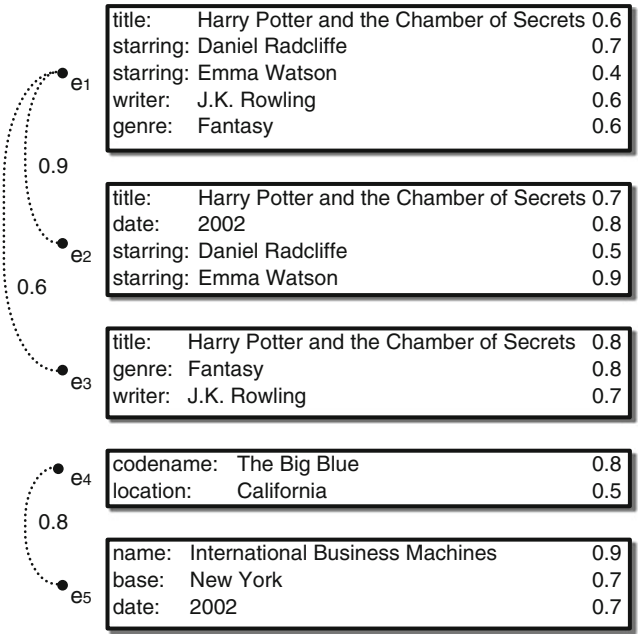

Fig. 9.2 Entities and their probabilistic linkage information, which is shown with the *dotted lines*

Consider now a user looking for the IBM consulting corporation. As is typically the case in dataspaces [22], queries are expressed as a series of attribute name–value pairs. Thus, the user sends the following query to the system:

$$\langle \text{name} = \text{“International Business Machines”}, \text{base} = \text{“New York”} \rangle$$

Clearly among the five entities e_1, e_2, e_3, e_4, and e_5, only the fourth satisfies these two conditions. Of course, since the attributes of the specific entity exist with some uncertainty, specified by the respective probabilities, the existence of the entity in the query answers should also be probabilistic. A significant amount of research has been carried out in the area of the probabilistic databases [12] on specifying the semantics and on the development of efficient query answering techniques for this kind of scenarios.

Assume now that a user is interested in the works of J.K. Rowling in the year 2002. He sends to the system the following query:

$$\langle \text{writer} = \text{“J.K. Rowling”}, \text{year} = \text{“2002”} \rangle$$

None of the three entities in Fig. 9.2 contain both attribute name–value pairs as specified in the query; thus, any probabilistic database approach will return an empty set as an answer. However, the linkage information between entity e_1 and e_2 indicates that they may represent the same real-world object. If they do, then they can be both merged into one entity, say e_{12} that contains as attributes the union of the attributes e_1 and e_2. That entity will satisfy both the conditions of the last query and should be part of the answer set, even though it is not one of the three entities that are actually stored in the repository.

In a similar situation, assume that the user sends the query:

$$\langle \text{writer} = \text{“J.K. Rowling”}, \text{genre} = \text{“Fantasy”} \rangle$$

Answer to the query should take into consideration all the different cases that may exist based on the entity linkages. In particular, a complete answer should contain three entities, namely, entity e_1, entity e_3, and the entity 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*.

The answer set for this query could also contain entities e_{12} and e_{123}, which are created by the merging of entity e_2 with e_1, and entity e_2 with e_1 and e_3, respectively. Included in the merging, the attributes of e_2 will create entities that have additional attributes, such as $date = 2002$. We consider such additional attributes as redundant, since the user did not request them through the query. Our basic principle is that we do not want to produce results that are not required, and therefore, no merging should take place unless it is justified by the query given by the user, the linkages, and the attributes composing the entities.

Our approach creates the entity mergings by using available entity linkages. Since the linkages are probabilistic, for an effective query mechanism, we need to take into consideration all the different combinations that may occur. Each such combination will partially contribute to the answer set. However, materialization of

all combination will lead to exponential increase of the data, which is inefficient to generate and store. Instead, query processing at runtime takes into consideration the related probabilistic linkages; computes the different combinations, along with their respective probability of existence; and then generates the answer set by merging the data produced from each combination.

9.1.4 Contributions

The main focus of this chapter is to efficiently and effectively address the entity linkage problem as this appears in heterogeneous, uncertain, and volatile data. This is achieved by allowing data integration systems to maintain probabilistic linkage information and perform entity-aware query processing over the data and thus retrieve answers to queries that reflect the corresponding real-world objects. This methodology avoids pitfalls that may result from the one-time a priori merging decisions, as performed by traditional entity linkage techniques. Furthermore, it can support highly volatile data more efficiently. The reason is that since no merging decisions have taken place, the only updates required are on the linkages related to new data incorporated in the system, or modified data.

In an effort to address the entity linkage problem for volatile data, we introduce a model for representing entities and linkages that aims at bringing together two worlds: the world of entity linkage and the world of probabilistic databases. The novelty of this data model is that it uses a generic entity-based representation model for highly heterogeneous data that support the simultaneous representation of possible linkages between entities alongside the original data, as generated by a number of the existing entity linkage techniques. This means that no data merging is performed in advance, but the outcome of the entity linkage algorithms, i.e., the pairs of entities possibly representing the same real-world object with the belief of that being true, is stored in the data. The outcome is a database that contains uncertainty not only on the attributes of the entities but also on their linkages.

Relying on the introduced model that contains a set of probabilistic linkages, we introduce a methodology to efficiently compute the answers for entity queries . Query answers reflect the entity linkage and entity representation information, with special emphasis given to the computation of the probabilities of the possible worlds based on the data and the matching uncertainty.

Generating entities by combining probabilistic linkages has several benefits. First, it produces additional valid query answering results compared to those of entity linkage and probabilistic databases, which cannot be simulated with previous techniques. An interesting feature is that reasoning about the entity linkages is done on the fly, meaning 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, i.e., by considering the union of all the attributes of the structures to be merged with corresponding probabilities [36].

9.1.5 Organization

The remaining of this chapter is structured as follows. Section 9.2 presents and discusses existing approaches that are related to the techniques presented in this chapter. Section 9.3 introduces and explains the data model, which includes the representation of entities and linkages, as well as the mechanism for dealing with data uncertainty. Section 9.4 explains the mechanism for dealing with probabilistic linkage information through on-the-fly entity-aware query processing. Section 9.5 reports on our experimental evaluation performed on two real-world datasets. Finally, Sect. 9.6 provides conclusions and provides an overview of current and future work.

9.2 Related Work

Most existing techniques for entity linkage focus on the off-line detection and linkage of data referring to the same real-world objects [14,19,20]. These techniques deploy a variety of different methodologies and directions. These includes string similarity metrics [8, 9] for computing the matching being the given textual representations of entities, the use of the available inner-relationships between the entities [17, 27], clustering [7], and blocking techniques [30, 31] for reducing the required execution time. However, as already explained in Sect. 9.1, these techniques have major limitations in addressing the entity linkage problem as this appears in current data [19], e.g., data evolution, uncertainty, and incompleteness.

Few existing data integration proposals focus on dealing with uncertain linkage information during query processing. More specifically, Dong et al. [18] 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 Sect. 9.1.1 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 will be 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 we know that some of the entities are the same, and thus they should be merged accordantly. In fact, the probabilistic schema mappings described in this approach could actually become an input to our approach.

The approach presented in [3] is more similar to ours, since their 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 (Sect. 9.1.1), this means that all tuples which describe the same entity should have the same identifier, e.g.,:

Identifier	Entity	Probability
5	e_1	p_1
5	e_2	p_2
5	e_3	p_3

The tuples that represent different entities are considered as independent and the tuples representing the same entity (having the same identifier) as conditionally dependent. The latter means that only one tuple for each identifier can be part of the results. Our proposal does not require this. We explain how entity linkages can contain correlations and provide an appropriate solution.

Other related approaches are dataspaces [22] and Trio [2]. The main focus of these approaches is to create database systems that support uncertainty along with inconsistency and lineage. At some extent, 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. Dalvis and Suciu [11] 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. [34] moved towards defining and using different correlations, e.g., that existence of one tuple implies or disallows the existence of another tuple.

9.3 Data Model

To effectively model highly heterogeneous information, we need a simple and flexible model that will be able to represent relational, XML, RDF, and object-oriented data without significant loss of information. We have chosen to go with a graph-based model that is typically used in dataspaces [22]. The main component of the model is an entity which consists of a number of attributes describing its characteristics and a set of associations between the entities. In particular, we assume the existence of an infinite set of entity identifiers $\mathcal{O}$, names $\mathcal{N}$, and atomic values $\mathcal{V}$. An entity is a design artifact used to model a real-world object. It consists of a unique entity identifier and a set of attributes. An *attribute* is a pair $\langle n, v \rangle$ of a name and a value and describes some characteristic of the entity. The set $\mathcal{A} = \mathcal{N} \times \mathcal{V}$ represents the infinite set of all the possible attributes.

Definition 9.1. An *entity* e is a tuple $\langle id, A \rangle$ called the *entity identifier* of the entity and $A \subseteq \mathcal{A}$ is a finite set called the set of *entity attributes*. ∎

Since each entity is distinguished by its unique identifier, for the rest of the document, the term entity and entity identifier will be used interchangeably.

Entity identifiers can be considered a special type of atomic values, allowing identifiers to serve as the value of an attribute. Through this mechanism, the data model is able to support not only entities and their characteristics but also relationships among them.

A database is a set of entities. Among these entities, there may be groups modeling the same real-world object but using a different or overlapping set of attributes. Such entities are said to be *linked*.

Definition 9.2. A *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, 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* and is denoted as $e_1 \equiv e_2$ if $(e_1, e_2) \in \mathcal{L}$. A maximal group of entities that are pairwise linked forms a *factor*. ∎

It is important to note that a linkage assignment can be equivalently expressed either through an explicit statement of the binary relationships or through a set of groups of entities, with each such group representing a factor. For instance, given six entities $e_1, e_2, \ldots, e_6$, the set $\{\{e_1, e_2, e_3\}, \{e_4, e_5\}, \{e_6\}\}$ describes a linkage assignment with three factors. The first factor consists of entities e_1, e_2, and e_3; the second contains e_4 and e_5; and the third contains only e_6. The set of linkages are the set of all the pairwise links in each factor.

Since linked entities in a database represent the same real-world object, they can be replaced by a new entity that combines the information described by them. (Note that linked entities may have different, or even disjoint, sets of attributes) This process is referred to as *entity merge* and leads to more compact database representations without losing any information. A database in which no merge can be performed is said to be *minimal*.

Definition 9.3. The *merge* of a set of entities $e_i = \langle id_i, A_i \rangle$ for $1 \leq i \leq n$, denoted as $merge(e_1, e_2, \ldots, e_n)$, is a new entity $\langle id, A \rangle$ such that id is a new identifier and $A = \cup_{i=1}^{n} A_i$. The *minimal* form of a database $\langle \mathcal{E}, \mathcal{L} \rangle$ is a database $\langle \mathcal{E}', \mathcal{L}' \rangle$, where $\mathcal{L}' = \emptyset$ and $\mathcal{E}' = \{e | e = merge(e_1, e_2, \ldots, e_n) \wedge \{e_1, e_2, \ldots, e_n\}$ is a factor in $\langle \mathcal{E}, \mathcal{L} \rangle\}$. ∎

To capture the uncertainty that may exist on the data, every attribute of an entity is associated with a value between 0 and 1, which indicates a likelihood that the information described by the attribute is among the characteristics of the real-world object that the entity models. Uncertainty exists also on the linkage information among the entities. Thus, we extend the definition of the database to include this uncertainty.

Definition 9.4. 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 and $\mathcal{L}$ is a linkage assignment on $\mathcal{E}$. p^a is a function that assigns a probability weight to the attributes of the entities, i.e., $p^a | B \mapsto [0, 1]$ with $B = \{a | a \in A \wedge \langle id, A \rangle \in \mathcal{E}\}$. p^l is also a function that assigns a probability weight to the entity linkages, i.e., $p^l | \mathcal{L} \mapsto [0, 1]$. ∎

Note, that our notion of a probabilistic database goes beyond the traditional probabilistic database [12] that simply associates probabilities with attributes, by assigning probabilities also to linkage relationships that exist among the entities.

Example 9.1. Figure 9.2 illustrates a small fraction of a probabilistic linkage database. It contains a total of five entities; therefore, $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$. The entity linkage techniques we executed on these five entities generated three entity linkages, and thus, $\mathcal{L} = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_4,e_5}\}$. From the figure, we can also see the attributes composing the entities. For example, entity e_1 has five attributes, with the first attribute having a value "Harry Potter and the Chamber of Secrets" for the name "title". The probability for the specific attribute is 0.6, which corresponds to the belief we have that the specific attribute describes e_1. These probabilities are provided by function p^a. Similarly, function p^l provides the probabilities for the entity linkages, shown in the figure with the line between the entities. ∎

Having the right probabilities on the attributes of a probabilistic entity is a critical issue. One of the challenges is to understand the semantics of these probability numbers and to adequately use them to perform the merge. This task is challenging since very often, different algorithms may have been used to compute the probabilities for the attributes of different entities, or the employed algorithms may not be known. In certain cases, these numbers may not even be probabilities in the strict mathematical sense but rather numbers that are meant to provide a relative ranking of the likelihood of the respective attributes in the entity. This can make the computation of the query results even harder. The same issues arise on the linkage information. There is a large amount of literature on the topic of computing entity linkage [19], with most methods analyzing the structural similarity of the data and returning some numbers measuring the likelihood that two data structures represent the same real-world entity. All these issues are outside of the scope of the current paper. We assume that this information is explicitly provided or computed in advance using some data analysis tools [6].

A probabilistic linkage database models multiple different real-world situations, i.e., possible databases, depending on what entity linkages actually exist among the entities and on what attributes each entity actually has. To differentiate between the situations that arise from the different linkages, the notion of *possible l-worlds* is introduced (short for possible linkage worlds).

Definition 9.5. Given a probabilistic linkage database $\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$, a *consistent* linkage specification is a linkage assignment $\mathcal{L}^{\text{sp}}$ such that $\forall x, y \in \mathcal{L}^{\text{sp}}$: $x, y \in \mathcal{L} \wedge p^l(x, y)! = 0$. The probabilistic database with linkages $\langle \mathcal{E}, \mathcal{L}^{\text{sp}}, p^a, p^l_{sp} \rangle$, where $p^l_{sp}(x, y) = 1$, $\forall (x, y) \in \mathcal{L}^{\text{sp}}$, is called a *possible l-world*. The set of all the possible l-worlds of probabilistic database with linkages $\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$ is denoted as $plw(\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle)$. ∎

Since a linkage specification uniquely defines a possible l-world (Definition 9.5), the terms "consistent linkage specification" and "possible l-world" will be used equivalently. Furthermore, we will consider only consistent linkage specifications; thus, we will not mention the word "consistent" any more.

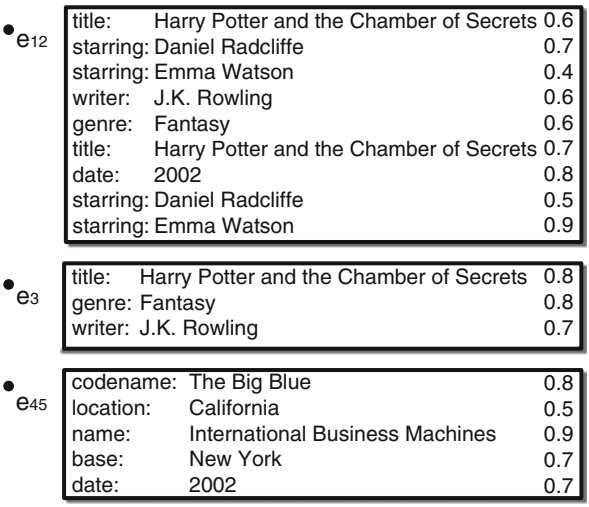

Fig. 9.3 An illustration of the linkage specification $\mathcal{L}^{\text{sp}} = \{l_{e_1,e_2}, l_{e_4,e_5}\}$ for the probabilistic linkage database shown in Fig. 9.2

Table 9.1 A summary of the notation introduced in Sect. 9.3 and used throughout this chapter

Notation	Description
$a_i = \langle n, v \rangle$	Attribute: a pair of name n and value v
$e_i = \langle id, A \rangle$	Entity: a tuple with an identifier id and a set of attributes A
l_{e_i,e_j}	Linkage: denotes a possible match between entity e_i with entity e_j
$\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$	Probabilistic linkage database
$\text{plw}(\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle)$	Possible l-worlds of a probabilistic linkage database
$\mathcal{L}^{\text{sp}}$	Linkage assignment
$\underset{i}{f}$	Factor: a set pairwise linked entities

Example 9.2. Consider again the probabilistic linkage database of Fig. 9.2. The entity linkage set is $\mathcal{L} = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_4,e_5}\}$. One of the possible linkage specifications $\mathcal{L}^{\text{sp}} = \{l_{e_1,e_2}, l_{e_4,e_5}\}$, which means accepting the two out of the three linkages of this original entity linkage set $\mathcal{L}$. As explained, each linkage we accept implies a merge between the entities of the linkages. Therefore, the specific linkage specification means we need to merge e_1 with e_2 and e_4 with e_5. The result is a probabilistic database with linkages, as shown in Fig. 9.3. ∎

In the remaining chapter, we will use the notation l_{e_1,e_2} as a shorthand for $(e_1, e_2) \in \mathcal{L}$. A summary of the introduced notation, which is used throughout this chapter, is listed in Table 9.1.

Note that a possible l-world still has probabilities assigned to the attributes of its elements. The possible worlds of each possible l-world describe a number of non-probabilistic databases, depending on whether each probabilistic attribute is present or not. These nonprobabilistic databases are referred to as *regular database*, or a *solution*. A solution actually corresponds to what is referred to as a possible world in the probabilistic database literature [12].

For queries, a simple, but flexible, query language is adopted. A query is a series of attributes, i.e., a list of name–value pairs. Intuitively, the semantics of the query is to discover entities that contain the attributes described in its attribute list. An entity e is included in the answer of a query q if it contains all the requested attributes. Evaluating the query on a probabilistic linkage database can be performed on the basis of the traditional query answering approaches.

Using entity linkage, it is possible to go beyond this expressiveness and be able to retrieve new entities that may not be explicitly represented in the database. These entities result from the merge of two or more entities as specified by their linkage information. In particular, the results of evaluating a query q over a probabilistic linkage database with a linkage $\mathcal{L}$ is equal to the results of the evaluation of the query q over the minimal form of the database, as specified by the linkage assignment $\mathcal{L}$.

If the database is probabilistic, and a specific linkage assignment $\mathcal{L}$ has been decided, then its minimal form can be computed but will also be probabilistic, since the entity attributes will have probabilities. Evaluation of a query over such a database can be performed based on various techniques proposed for probabilistic databases [12]. However, if the linkage assignment is also probabilistic, as in the general case of a probabilistic database, then a query is evaluated over the database by computing all its possible worlds, i.e., all the possible linkage assignments, and then evaluating the query on the minimal form of each such worlds. The final result of the query will by the union of the results of the evaluations on the individual worlds. However, since the linkage information is probabilistic in the first place, so is the linkage assignment, and as a consequence, the results of each evaluation on the individual worlds should also be coming with some probability.

A fundamental property that differentiates our work from other work related to querying probabilistic data is that the query results are computed from entities that are compiled on the fly at query execution time from the available linkage information. The entities used in query evaluation thus consider entities that are not materialized in the repository in this form. In particular, traditional approaches evaluate the queries on the extensional data that can be found in the database. In our work, we view the probabilistic entity linkage as an intensional description of a number of possible entities that can result from the merge of those linked. We compute these entities through the possible worlds on the fly and offer additional query results.

Example 9.3. Consider again the probabilistic linkage dataset illustrated in Fig. 9.2 and the query:

⟨starring:“Emma Watson”, starring: “Radcliffe, Daniel” ⟩

In can be observed that there is no entity with both the attributes requested by the query. Traditional query answering techniques would have failed to return results or would have returned with a low confidence, those having at least one of the two requested attributes. However, in the possible world described by the entity linkage $\{\{e_1, e_2, e_3, e_4, e_6\}, \{e_8, e_9\}\}$, the merge of the entities of the first of the two factors represents an entity with both the attributes requested in the query. Thus, such an entity can be returned with a much higher confidence. ∎

The following sections deal with the challenge of computing the right probabilities of the possible l-worlds and possible worlds, and performing query answering on the fly without having to materialize all them.

9.4 Efficient Query Evaluation

In this section, we present how query evaluation can be performed efficiently over a probabilistic linkage database. A more detailed description of the algorithm and experimental evaluation is available in [24] and [25].

Our query evaluation approach is based on an idea similar to the one incorporated in probabilistic databases for dealing with datasets of large sizes. In particular, the probability values on the linkages are interpreted as the probability distribution over the set of all the possible l-worlds $plw(\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle)$. Given a linkage specification, i.e., a possible l-world, the probability values of the attributes are interpreted as the probability distribution over all the possible solutions (ref. Sect. 9.3) within one possible world leading to a two-level approach. The answer of a query on a probabilistic linkage database, thus, is the union of the answers over all the possible worlds of all the l-worlds fulfilling the query conditions. Each element in the answer set, however, is accompanied by an aggregated probability value which reflects the probability of existence of the specific possible world, which contains the answer element, as well as the probability of the possible l-world, which contains the respective solution.

This description of semantics for query answering is indirectly suggesting a query evaluation strategy: compute all possible worlds of all possible l-worlds, compute their probability of existence, evaluate the query in each one of them, and return the results along with computed probability. Unfortunately, following this approach is prohibitively expensive.

Here, we propose an alternative evaluation strategy that avoids the high computational cost without any loss of accuracy. The basic idea is to restrict the computation to only those possible l-worlds that are meaningful for the query at hand. Since there is a one-to-one correspondence between possible l-worlds and linkage specifications, and between linkage specifications and entity merges, we start from the entity merges that are required in order to generate an answer to the given query. From the merges, we can find the linkage specifications, and from these assignments, the possible l-worlds. The probability of each possible l-world is computed based on the probabilities of the linkages included or not included when generating the specific l-world. Finally, the possible worlds of each l-world are generated along with their own probability, which is combined with the probability of the respective world and then included in the answer set of the query. An overview of the proposed query evaluation is given by Algorithm 18, while the following subsections describe these steps of the algorithm in more detail.

Algorithm 18: Query evaluation

```
Input:  Q: a query describing an entity
Output: R: a set of entities satisfying query conditions
LS ← findRequiredLinkageSpefications(Q);
PLW ← ∅;
R ← ∅;
foreach ls ∈ LS do
    W ← findPossibleLWorlds(ls);
    foreach w ∈ W do
        w.prob ← calculateWorldLProbability(ls);
        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};
```

9.4.1 Indexing Structure

A commonly used approach in answering queries over probabilistic data is to partition the data into a series of disjoint/independent groups [4, 12, 32, 34]. These groups can be found in the literature under the name factors [34] or components [4]. The set of the possible combinations between the data of these groups produces all the possible worlds.

This idea is not directly applicable to our case. The reason is that the existing approaches operate under the assumption that the data within one factor or component are independent. This assumption does not hold in our case. The transitive property of the linkages may generate additional correlation, i.e., dependencies, that are equally important for the correct identification of the possible worlds. For instance, entity linkages l_{e_1,e_2} and l_{e_1,e_3} without linkage l_{e_2,e_3} cannot be considered, since the first two linkages imply the information encoded in the third linkage.

We follow an idea similar to the management of uncertain data with correlations [34]. As a first step, we divide the set of entities into sets of connected components, i.e., factors (Definition 9.2). For example, given $\mathcal{L} = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_2,e_3}, l_{e_8,e_9}\}$, two independent factors can be identified. The first factor contains entities e_1, e_2, and e_3 with linkages $L_1 = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_2,e_3}\}$, while the second factor contains simply e_8 and e_9 with one linkage $L_2 = \{l_{e_8,e_9}\}$.

To compute all possible l-worlds of a probabilistic linkage database $\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$, we need to consider all the possible valid linkage specifications of $\mathcal{L}$. This number can easily get large enough 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. We will use notation $\mathcal{L}^{\text{sp}}_{f_i}$ to denote all the possible valid linkage specifications between entities of the ith factor and $\mathcal{L}^{\text{sp}}_{f_i}(\text{k})$ one of the possible assignments.

Each possible l-world is using a specific specification within each factor. Thus, the set of possible l-worlds can be derived by combining all the alternative valid linkage specifications within each factor as follows:

$$\text{plw}(\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle) = \mathcal{L}^{\text{sp}}_{f_1} \times \mathcal{L}^{\text{sp}}_{f_2} \times \cdots \times \mathcal{L}^{\text{sp}}_{f_n}$$

Example 9.4. Consider a probabilistic linkage database with entity linkages $\mathcal{L} = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_4,e_5}\}$. For the specific set of entity linkages, we have two factors. Factor f_1 with entities $\{e_1, e_2, e_3\}$ and factor f_2 with entities $\{e_4, e_5\}$. The corresponding linkage specifications are $\mathcal{L}^{\text{sp}}_{f_1} = \{l_{e_1,e_2}, l_{e_1,e_3}\}$ and $\mathcal{L}^{\text{sp}}_{f_2} = \{l_{e_4,e_5}\}$, and thus, the possible l-worlds can be retrieved as follows:

$$\begin{array}{lcl}
\mathcal{L}^{\text{sp}}_{f_1}(1) = \{l_{e_1,e_2}, l_{e_1,e_3}\} & & \mathcal{L}^{\text{sp}}_{f_2}(1) = \{l_{e_4,e_5}\} \\
\mathcal{L}^{\text{sp}}_{f_1}(2) = \{l_{e_1,e_2}\} & \times & \mathcal{L}^{\text{sp}}_{f_2}(2) = \{\} \\
\mathcal{L}^{\text{sp}}_{f_1}(3) = \{l_{e_1,e_3}\} & & \\
\mathcal{L}^{\text{sp}}_{f_1}(4) = \{\} & &
\end{array}$$

The cartesian product of these linkage assignments results into the number of possible l-worlds for the whole database. The next table illustrates these l-worlds (through the specifications) along with the entity merges for each of these possible l-world.

Possible worlds	Entity merges
$D_1 = \{l_{e_1,e_2}, l_{e_1,e_3}, l_{e_4,e_5}\}$	$e_1 \equiv e_2 \equiv e_3, \quad e_8 \equiv e_9$
$D_2 = \{l_{e_1,e_2}, l_{e_1,e_3}\}$	$e_1 \equiv e_2 \equiv e_3, \quad e_8, \quad e_9$
$D_3 = \{l_{e_1,e_2}, l_{e_4,e_5}\}$	$e_1 \equiv e_2, \quad e_3, \quad e_8 \equiv e_9$
$D_4 = \{l_{e_1,e_2}\}$	$e_1 \equiv e_2, \quad e_3, \quad e_4, \quad e_5$
$D_5 = \{l_{e_1,e_3}, l_{e_4,e_5}\}$	$e_2, \quad e_1 \equiv e_3, \quad e_4 \equiv e_5$
$D_6 = \{l_{e_1,e_3}\}$	$e_2, \quad e_1 \equiv e_3, \quad e_4, \quad e_5$
$D_7 = \{l_{e_4,e_5}\}$	$e_1, \quad e_2, \quad e_3, \quad e_4 \equiv e_5$
$D_8 = \{\}$	$e_1, \quad e_2, \quad e_3, \quad e_4, \quad e_5$

■

To avoid recomputing the factors every time, we maintain an index structure which is dynamically maintained. The index structure is based on the idea of equivalence classes. In reality, each factor is actually an equivalence class. When data are modified and new linkages are introduced or old ones are eliminated, changes should occur in the equivalence class memberships, thus on the factors. Algorithm 19 illustrates how the factor index is maintained under new linkage insertions.

Once the various l-worlds have been constructed, the second important step is to compute the probability of each possible l-world. Since the factors are independent of each other, the probability of an l-world can be computed as the product of the probabilities of the involved factors (Sect. 9.4.3). It is thus given by:

Algorithm 19: Updating factor indexing

Input: (a) $l_{\alpha,\beta} := (e_\alpha, e_\beta, p)$: a new linkage
(b) $F := F_1, F_2, \ldots, F_n$: the factors
Output: Updated factors F
$F_\alpha \leftarrow$ getFactorOf(e_α);
$F_\beta \leftarrow$ getFactorOf(e_β);
if $F_\alpha \notin F$ **then**
  $F \leftarrow F \cup F_\alpha$;
if $F_\beta \notin F$ **then**
  $F \leftarrow F \cup F_\beta$;
if $F_\alpha == F_\beta$ **then**
  // already in the same factor ;
  return F;
else if F_α != F_β **then**
  $F_\gamma \leftarrow F_\alpha \cup F_\beta$;
  $F \leftarrow F \setminus F_\alpha \setminus F_\beta \cup F_\gamma$;

$$\Pr(\text{l} - \text{world}) \;=\; \prod_{i=1}^{n} \Pr(\mathcal{L}^{\text{sp}}_{\text{f}_i}(.)) \tag{9.1}$$

Example 9.5. Assume that the possible worlds that satisfy condition $e_1 \equiv e_2$ and $e_4 \equiv e_5$ needs to be retrieved. Based on Example 9.4, it can be easily seen that the left part of the condition is satisfied by factors $\mathcal{L}^{\text{sp}}_{\text{f}_1}(1)$ and $\mathcal{L}^{\text{sp}}_{\text{f}_1}(3)$, while the right part by factor $\mathcal{L}^{\text{sp}}_{\text{f}_2}(1)$. Therefore, only the possible l-worlds given by the product of these factors needs to be considered. These are:

$$\begin{aligned} &\mathcal{L}^{\text{sp}}_{\text{f}_1}(1) = \{l_{e_1,e_2}, l_{e_1,e_3}\} \qquad \times \qquad \mathcal{L}^{\text{sp}}_{\text{f}_2}(1) = \{l_{e_4,e_5}\} \\ &\mathcal{L}^{\text{sp}}_{\text{f}_1}(3) = \{l_{e_1,e_3}\} \end{aligned}$$

This results in two possible l-worlds. The first l-world is given by

$$\text{plw}_1 = \mathcal{L}^{\text{sp}}_{\text{f}_1}(1) \times \mathcal{L}^{\text{sp}}_{\text{f}_2}(1) \text{ with probability } \Pr(\mathcal{L}^{\text{sp}}_{\text{f}_1}(1)) \cdot \Pr(\mathcal{L}^{\text{sp}}_{\text{f}_2}(1)) \tag{9.2}$$

and the second l-world is given by

$$\text{plw}_2 = \mathcal{L}^{\text{sp}}_{\text{f}_1}(3) \times \mathcal{L}^{\text{sp}}_{\text{f}_2}(1) \text{ with probability } \Pr(\mathcal{L}^{\text{sp}}_{\text{f}_1}(3)) \cdot \Pr(\mathcal{L}^{\text{sp}}_{\text{f}_2}(1))$$

■

9.4.2 Retrieving Possible l-Worlds

In order to avoid the creation of all possible l-worlds when a query is issued, we exploit the list of factors we are maintaining, as mentioned in the previous

section, and the attributes in the query. We do so in order to restrict the creation of only those possible l-worlds that are necessary, i.e., those that will lead to the generation of some query results. We achieve this by first detecting the entity merges that are required in order to satisfy the conditions of the query. In particular, for every attribute specification a_i in the query, a list E_{a_i} is constructed with all the entities having attribute a_i. Clearly an entity satisfies all the query conditions, that is, contains all the attributes set by the query, if it appears in each one of the lists E_{a_i}, for $i = 1..n$. It is not always the case that such an entity exists. However, by merging two or more entities, it is possible to create a new one (the result of the merge) whose attributes contain all the attributes in the query. Let S be a set of such entities. For the set S to serve the required purpose, two main properties need to be satisfied. First, all its members have to belong to the same factor (it is not possible to talk about merge of entities that belong to different factors), and second, there must be at least one entity $e \in S$ from every list E_{a_i}, with $i = 1...n$. Of course, one can always create "super entities" by merging all the entities in every factor. However, this may generate merges that are not necessary. To avoid this form or redundancy, we require that each set E_{a_i} contributes at most one entity. This means that the number of entities to merge can never be more than the number of attributes in the query.

To compute all the possible merge combinations, we generate the cartesian product of the sets E_{a_i} with the extra requirement that they should belong to the same factor. Algorithm 20 provides a brief description of the described steps.

Example 9.6. For example, assume that a query q contains attributes a_1, a_2, and a_3, and each one is satisfied by the entity sets $E_{a_1} = \{f_1 - e_a, f_1 - e_b, f_2 - e_c\}$, $E_{a_2} = \{f_1 - e_b, f_2 - e_d\}$, and $E_{a_3} = \{f_1 - e_g, f_1 - e_h, f_2 - e_i\}$, respectively. For each entity in the sets, we also indicate the factor in which the entity belongs to. We first compute all the possible combinations:

$$\begin{array}{c} E_{a_1} \\ \hline f_1 - e_a \\ f_1 - e_b \\ f_2 - e_c \end{array} \times_{(E_{a_1}.f_i = E_{a_2}.f)_j} \begin{array}{c} E_{a_2} \\ \hline f_1 - e_b \\ f_2 - e_d \end{array} \times_{(E_{a_2}.f_i = E_{a_3}.f_j)} \begin{array}{c} E_{a_3} \\ \hline f_1 - e_g \\ f_1 - e_h \\ f_2 - e_i \end{array}$$

which lead to the following merges: $merge(e_a, e_b, e_g), merge(e_a, e_b, e_h)$, $merge(e_b, e_g), merge(e_b, e_h)$, and $merge(e_c, e_d, e_i)$. Note that e_b belongs to both sets E_{a_1} and E_{a_2}, which means that the entity e_b has both attributes a_1 and a_2; thus, being merged with e_g or e_h is enough to create an entity that satisfies the query conditions. ∎

Consider now a merge of entities e_α and e_β from a factor f_i. From all possible worlds that can be generated from the specific factor, we are only interested in those that contain the specific entity merge. The remaining can be ignored. All entities constructed from this merge will contain some common attributes, which correspond to the attributes directly coming from the entities e_α and e_β. The

Algorithm 20: Generate entity merges

Input: $Q := \langle a_1, a_2, \ldots, a_k \rangle$: an entity query
$\langle \mathcal{E}, \mathcal{L}, p^a, p^l \rangle$
: a probabilistic linkage database**Output**: M: a set of entity merges
foreach a_i *in* Q **do**
  $E_i \leftarrow \{e \mid e = \langle id, A \rangle \wedge e \in \mathcal{E} \wedge a_i \in A\}$;
$N \leftarrow \{(e_1, \ldots e_n) \mid \forall i = 1..n : e_i \in E_i \wedge \forall i = 2..n : factor(e_{i-1}) = factor(e_i)\}$;
$M \leftarrow \{eliminateDuplicates(m) \mid m \in N\}$;

remaining attributes in the possible l-worlds will come from entities participating in other linkages which exist in the specific possible l-world. To further optimize our framework, we exploit this behavior and instead of returning all the entities for each factor, we return only one *partial entity*. The partial entity contains the minimum set of attributes required by the specific merge since this is commonly found in all entities generated by this factor.

9.4.3 Computing Probabilities of Possible l-Worlds

The next step in the query answering process is to decide the probability of a merge, i.e., a partial match as mentioned in the previous section. Recall that a partial match may be true in many possible worlds. There are two alternatives that one can follow. The first is to assign as the probability of the partial match the sum of the probabilities of these worlds. 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 world with the highest probability. For systems that simply use the match probability as a ranking mechanism for the entities before displaying them to the user, this second option is typically sufficient.

The algorithm for computing the match probability is based on the algorithm for finding shortest paths on graphs. In particular, provided the entity linkages L_i in a factor, we generate a weighted undirected graph G as follows: every entity participating in the linkages of L_i becomes a node of the graph. Each linkage l_{e_α, e_β} becomes an edge that connects the nodes representing entities e_α and e_β. The weight of such an edge is given by the probability of the respective linkage.

An entity merging $merge(e_1, e_2, \ldots, e_n)$ corresponds to a spanning tree that connects all entities $e_1, e_2, \ldots, e_n$. Computing the merge 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 a

list of edges is considered the subset of the edges that have one endpoint marked visited and one nonvisited. The one with the highest probability is selected and its nonvisited endpoint is marked as visited. The same step is repeatedly executed 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 this probability is actually the maximum.

9.4.4 *Retrieving and Computing Probabilities of Possible Worlds*

The previous sections have dealt with the problem of processing a query by efficiently deriving the possible l-worlds $\text{plw}(\langle \mathcal{E}, \mathcal{L}_c, p^a, p^l \rangle)$ along with the corresponding entity merges. However, this is not all. Recall that entity attributes themselves have probabilities; thus, even a possible l-world may be describing multiple different (nonprobabilistic) databases. These different databases are what we call solutions.

Each solution essentially represents a different combination over the attributes of the entities participating in a specific merge. For instance, consider the data in Fig. 9.3 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 Fig. 9.4. 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 simplest approach regarding attribute dependencies is to consider the attributes as independent and include them all as attributes in the merge result entity. However, 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 highly depend on the nature of the sources and their datasets. Our framework is able to handle such correlations in a uniform manner. A simple method is to cluster the exclusive attributes from each entity, i.e.,

Attributes for *merge*($e_1 \equiv e_2$)

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	Radcliffe, Daniel	0.5
◇ a_{21}	starring	Watson, Emma (II)	0.9

Solutions

(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}

Fig. 9.4 The attributes involved in $merge(e_1, e_2)$ along with the solutions when we deal with exclusive attributes (same name and similar values)

$M = \{\{e_1.\alpha_i, e_1.\alpha_j, \ldots\}\}$. We can then use this set to generate the solutions and compute their probability. The following paragraphs provide more details for generating solutions.

9.4.4.1 Independent Attributes

One option is to assume no correlation between the attributes and thus no restrictions on which attributes to include in the merge result entity. In this case, there is only one solution which is given by the union of all entity attributes.

$$\text{merge}(e_1, e_2, \ldots, e_n) = \langle id', \cup_{i=1}^{n} e_i.A \rangle$$

9.4.4.2 Exclusive Attributes

In certain cases, the attributes originating from different entities participating in the entity merge are exclusive. 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 is 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 Fig. 9.4. 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:

$$\text{merge}(e_1, \ldots, e_n) = \langle \text{id}', A \rangle, \text{where}$$

$$A \subseteq (M_1 \times M_2 \times \cdots \times M_m) \cup \{\alpha \mid \alpha \notin \cup_{i=1}^{m} M_i.\alpha\}.$$

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:

$$\Pr(e' \mid \text{merge}(e_1, \ldots, e_n)) = \Pr(l - \text{world}) \times \prod_{\alpha \in e'.A} p^\alpha \times \prod_{\alpha \notin e'.A \,\&\, \alpha \in e_i.A} (1 - p^\alpha).$$

Example 9.7. Figure 9.4 shows the attributes involved in *merge*(e_1, e_2). The exclusive attributes are given by set $M = \{\{\alpha_{10}, \alpha_{20}\}, \{\alpha_{11}, \alpha_{21}\}\}$. Figure 9.4 shows the four generated possible worlds, and their probability is computed according to above formula.

9.5 Evaluation

This section presents the results of the experimental evaluation for the suggested entity linkage methodology. The goal was twofold: (1) to study the effectiveness of our approach and identify its advantages over traditional techniques for entity linkage and (2) to investigate the efficiency of query processing and the overhead it introduces. We implemented our approach using Java 1.6 and performed all experiments on a computer with 5,400 rpm hard disk, a core 2 duo processor of 1.8 Ghz, and 2 GB RAM. For storing entities, we used MySQL 5, on the same computer. The following paragraphs present our datasets.

Movie Dataset. For evaluating the efficiency, we needed a sufficiently large dataset. Also, to investigate effectiveness, we needed linkages generated from different linkage techniques. We generated such a dataset by integrating data describing movies coming from two real-world systems, *IMDb* and *DBpedia*. *IMDb* data were stored in relational format, and *DBpedia* data were stored in RDF format. We converted both datasets to our data model and stored them in a relational database. To find the true matches (i.e., ground truth), we have used the *imdb_id* field from the *DBpedia* dataset, which contains the id of the movie in the *IMDb* dataset. The table in Fig. 9.5 shows the details for the movie dataset.

For generating the entity linkages, we compared the movie titles using two standard string similarity methods [9], *Jaccard* and *Jaro*. Figure 9.5 plots the precision-recall plot 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 clear dependency between precision and recall. While recall increases, precision decreases, and vice versa. The linkage techniques always have to decide the trade-off between precision and recall. In our experiments, we investigate how our approach addresses this issue.

IMDb		DBpedia		Real-world objects
Entities:	23.182	Entities:	28.040	13.435
Attributes:	820.999	Attributes:	186.655	

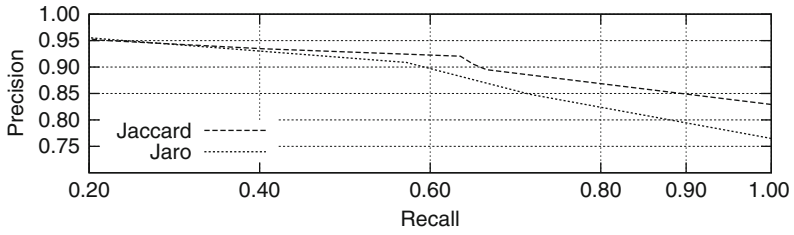

Fig. 9.5 Information about the movie datasets: the data composing the dataset (*top*) and the precision-recall plot for two entity linkage techniques (*bottom*)

Attributes	Entities	Real-world objects
5764	2882	9774

Entity linkages (under threshold t)					
$t = 0.52$	$t = 0.58$	$t = 0.62$	$t = 0.68$	$t = 0.72$	$t = 0.78$
12440	12012	10775	6394	5985	4184

Fig. 9.6 Information about the Cora datasets: the data composing the dataset (*left*) and the number of linkages for various thresholds (*right*)

Cora Dataset. Our second dataset was a collection of publications and authors from CiteSeer.[1] Cora dataset is typically used to evaluate entity linkage techniques [3, 16, 17]. Its data comes from CiteSeer, a real-world application, and it contains various author descriptions that refer to the same real-world object (maximum is 88 author instances describing the same real-world object).

We generated entity linkages between authors (i.e., entities) using the probabilistic entity linkage [26]. Entity linkage techniques typically select a linkage threshold and incorporate in the original data all entities with corresponding linkages above this threshold. Figure 9.6 provides the details for this dataset along with the number of linkages under different thresholds. Precision and recall of the generated entity linkages are similar to the ones generated by other algorithms such as [16, 17]. For our approach, we did not apply such a threshold but also used the linkages with low probabilities.

Entity Queries. The evaluations for both datasets were performed with 800 queries. Each query was generated by randomly selecting attributes of entities belonging to the same real-world object. We generated queries for real-world objects which contained at least two entities in our dataset. All reported results are computed on the average of 800 queries.

9.5.1 Effectiveness of Query Processing

We evaluate effectiveness of entity-aware query processing in a twofold manner. First, we examine the quality of entities returned when querying with our approach. Second, we compare entities returned from entity-aware query processing (EAQP) with entities returned when we use directly the results of the entity linkage techniques (ELA), i.e., applying the threshold on the entity linkages. We performed both evaluations using the Cora Dataset.

Our first experiment was as follows. We used the entity linkage technique to link all authors in the Cora dataset and stored the proposed linkages in the database. Then we processed the 800 queries and compared the results returned with EAQP and with ELA. As we already explained, entity linkage techniques select a threshold and accept linkages that have a higher probability than this threshold.

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

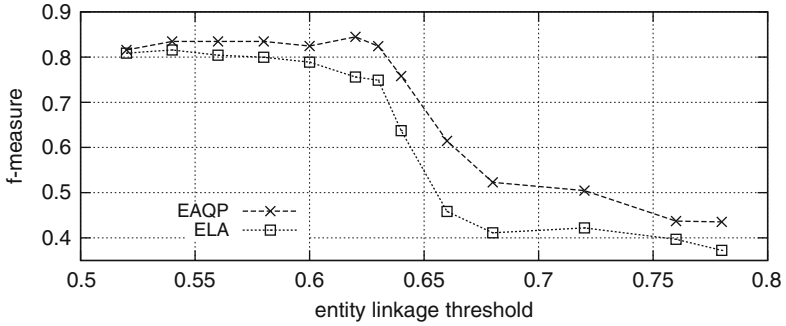

Fig. 9.7 F-measure with entity-aware query processing (EAQP) and entity linkage algorithm (ELA) over various thresholds for the accepted linkages

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 also with lower recall. One can easily see that when the selected threshold increases, the number of linkages is reduced since we then accept only the entities with high probabilities (see Fig. 9.6 for the exact numbers). We examine the behavior of the entity-aware query processing as well as the entity linkage algorithm when the value of the threshold is increased.

To measure quality of query results, we computed F-measure, which is a weighted harmonic mean of precision and recall. We consider a query as correct when it returns the real-world object by merging the information found in the various corresponding entities.

Figure 9.7 shows the average F-measure of the 800 queries for various entity linkage thresholds. As expected, when moving toward higher thresholds, the entity linkage technique accepts less and less linkages. This makes the technique unable to find the entities described by the queries. On the contrary, even for high thresholds, EAQP is able to identify the entities. For example, for $t = 0.66$, EAQP returned the correct entity for around 10% more queries than ELA. This is because EAQP can find connected linkages to construct the entity described in the query. ELA had to reject these linkages because of their low threshold. The exact precision and recall values for some of these thresholds are shown in the following table:

t	EAQP		ELA	
	Precision	Recall	Precision	Recall
0.62	1.00	0.73	1.00	0.61
0.63	0.99	0.71	1.00	0.60
0.64	0.91	0.65	1.00	0.47

Figure 9.8 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.

We further analyzed the results of this evaluation 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 for 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 Fig. 9.8, for $t = 0.6$, EAQP returned 421 correct answers, whereas ELA returned 238 correct answers. For 91 answers, EAQP had higher probability than ELA. Figure 9.9 presents the numbers for these two situations.

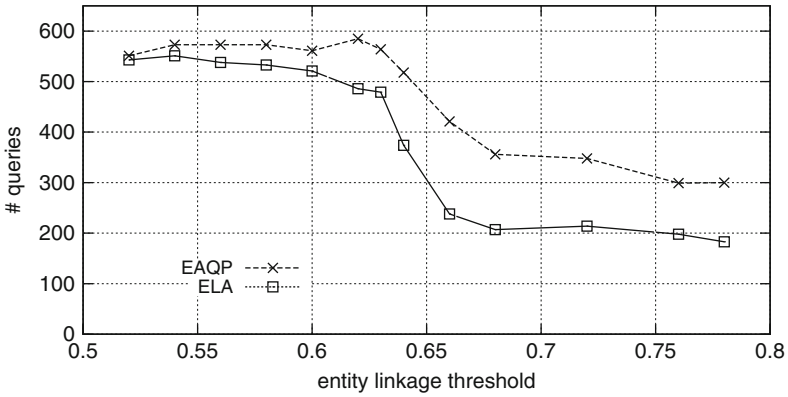

Fig. 9.8 Number of queries correctly returned with EAQP and ELA over various thresholds for the accepted linkages

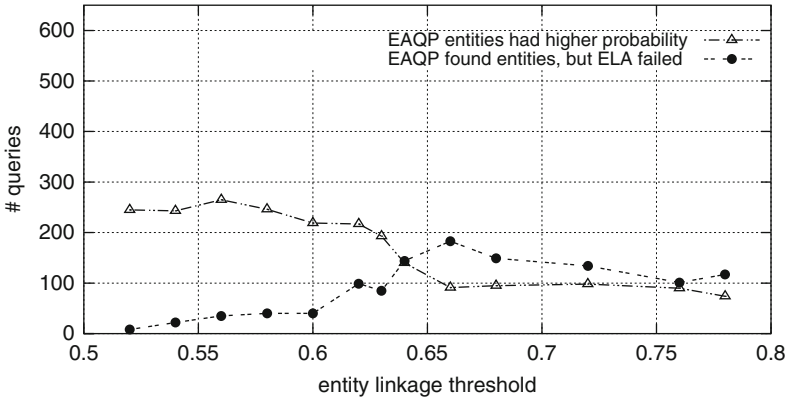

Fig. 9.9 Number of queries from the ones shown in Fig. 9.8 in which EAQP returned answers with higher probability and identified requested entities, whereas the ELA failed

9.5.2 *Efficiency of Query Processing*

We now report the results for the efficiency evaluation of our approach.

Size of Generated Factors. The core of our approach is based on generating factors by grouping linkages which are pairwise linked (Sect. 9.4.1). During query processing, we select the related factors and process them to construct the entities. The sizes of the factors influence the execution time of our approach.

We computed the size of the generated factors as the number of entity linkages contained in the factors. We then constructed the histogram of factor sizes. Figure 9.10 shows appearances per factor sizes as generated by both entity linkage techniques, *Jaro* and *Jaccard*. As shown, most of the factors have a small size and few factors are of bigger size. In addition, we see that for the entity linkage technique generated with *Jaro*, we have more factors of bigger sizes. Considering again the characteristics of the linkages generated by the *Jaro* and *Jaccard* (cf. Fig. 9.5), precision and recall results of *Jaccard* were better than *Jaro*. Clearly, *Jaro* is less capable to identify the correct linkages between entities, and thus, it generates more linkages which are less certain. This generates more pairwise-linked entities which are now reflected in the size of factors.

Time to Retrieve Possible Worlds. Given that factors have different sizes, we measured the time needed to identify the possible world with the highest probability in respect to the factor size (Sect. 9.4.3). Figure 9.11 shows the required time for different factor sizes. As expected, for small factor sizes (i.e., 20–40 entity linkages) which is the dominating majority among the factors, the algorithm requires around 1 millisecond. For larger factor sizes, the algorithm requires more time, which however still remains below 4 ms.

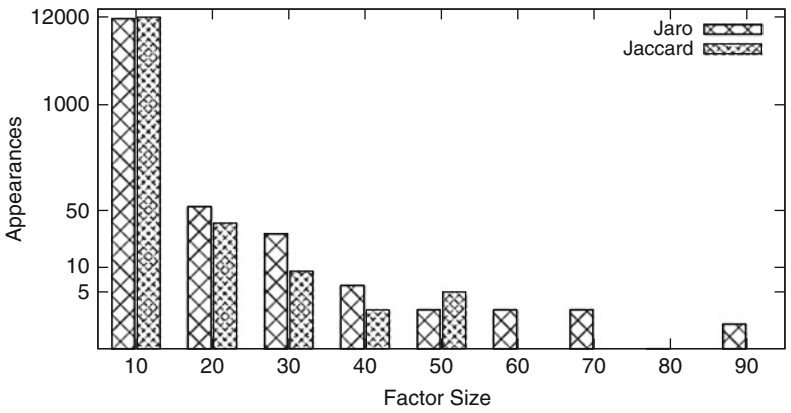

Fig. 9.10 Appearance numbers for the factor sizes generated for our two movie datasets

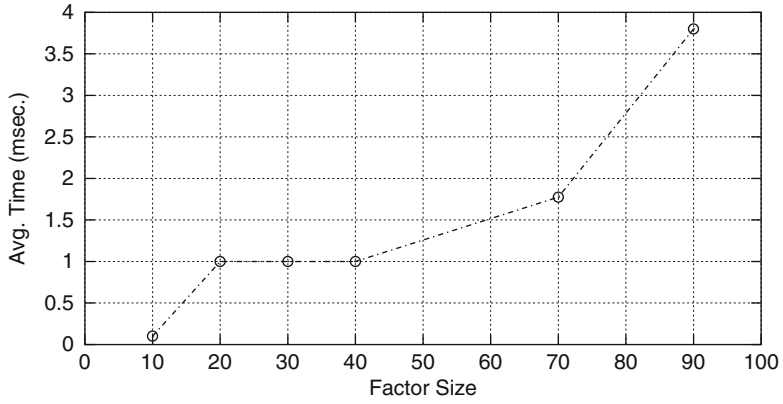

Fig. 9.11 Average time for computing the possible world with the maximum probability over factor sizes

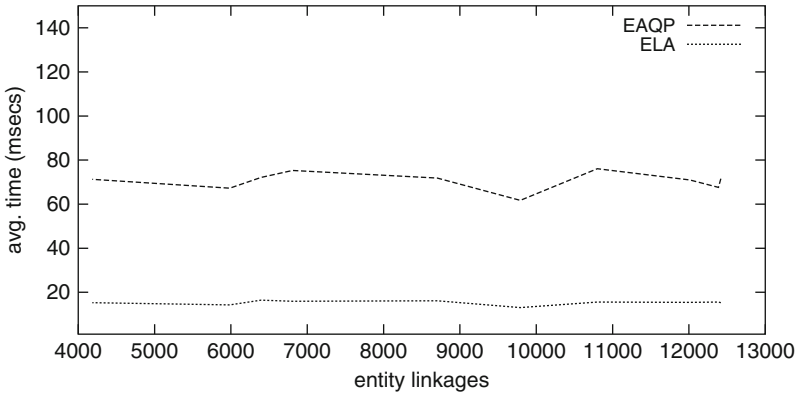

Fig. 9.12 Total time for processing queries over different numbers of entity linkages in dataset

Execution Time. Our final evaluation was to measure the time required for entity-aware query processing and also to compute the overhead that a system will have for offering this additional functionality. Figure 9.12 shows the average time taken to answer queries with and without our approach. We show time over different numbers of entity linkages in dataset. As expected, there is an increase in the time required with our approach, but this is relatively small and it remains under 70 ms. Furthermore, time does not increase as the dataset gets larger. On the contrary, query time remains stable even when the dataset is double the size. This behavior results from the effective grouping of linkages into factors which allows the algorithm to easily detect and use only a small subset of the linkages during query processing.

9.5.3 Evaluation Summary

Summarizing, the result of our experimental evaluation confirms the following:

- Incorporating entity-aware query processing in a system makes the system able to better handle the entity linkage problem and especially provide query answers which reflect the possible entity solutions for the current data.
- Our approach has a small overhead in time required for processing queries, but due to our efficient processing strategy, this cost remains low and constant even for large datasets with a large amount of entity linkages.

9.6 Conclusions

We have introduced a novel approach that allows on-the-fly entity-aware query processing in the presence of linkage information. Our approach can be applied on various data formats and structures using a generic entity representation. We explained how query processing can be performed efficiently over the entities and their possible linkages as these are generated by existing entity linkage techniques. Special focus was given on handling the uncertainty that appears in the entity linkage information as well as in the entity data. Our evaluation shows that the approach is both efficient and effective in answering entity queries.

We are currently investigating several directions to extend our approach. First, we would like to cover provenance information related to the possible linkage decisions and answers returned by querying. Also, we would like to investigate the implications of having conflicting information for the entity descriptions, as is typically the case for Web data, and to try out effective ways to deal with such conflicts.

References

1. Adar, E., Re, C.: Managing uncertainty in social networks. IEEE Data Eng. Bull. 15–22 (2007)
2. Agrawal, P., Benjelloun, O., Sarma, A., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: a system for data, uncertainty, and lineage. VLDB, pp. 1151–1154 (2006)
3. Andritsos, P., Fuxman, A., Miller, R.: Clean answers over dirty databases: a probabilistic approach. ICDE (2006)
4. Antova, L., Koch, C., Olteanu, D.: $10^{(10)^6}$ worlds and beyond: efficient representation and processing of incomplete information. VLDB J. **18**(5), 1021–1040 (2009)
5. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S., Widomr, J., Jonas, J.: Swoosh: a generic approach to entity resolution. VLDB J. **18**(1), 255–276 (2009)
6. Bex, G., Neven, F., Vansummeren, S.: Inferring xml schema definitions from xml data. VLDB, pp. 998–1009 (2007)
7. Bhattacharya, I., Getoor, L.: Iterative record linkage for cleaning and integration. DMKD, pp. 11–18 (2004)

8. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name matching in information integration. IEEE Intel. Syst. **18**(5), 16–23 (2003)
9. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string distance metrics for name-matching tasks. IIWeb, pp. 73–78 (2003)
10. Dalvi, N., Kumar, R., Pang, B., Ramakrishnan, R., Tomkins, A., Bohannon, P., Keerthi, S., Merugu, S.: A web of concepts. PODS, pp. 1–12 (2009)
11. Dalvi, N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB J. **16**(4), 523–544 (2007)
12. Dalvi, N., Suciu, D.: Management of probabilistic data: foundations and challenges. PODS, pp. 1–12 (2007)
13. Dasu, T., Johnson, T.: Exploratory Data Mining and Data Cleaning. Wiley, NY, USA (2003)
14. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: a brief survey. AI Mag. **26**(1), 83–94 (2005)
15. Doan, A., Lu, Y., Lee, Y., Han, J.: Object matching for information integration: a profiler-based approach. IIWeb, pp. 53–58 (2003)
16. Domingos, P.: Multi-relational record linkage. Multi-relational data mining workshop co-located with KDD, pp. 31–48 (2004)
17. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. SIGMOD conference, pp. 85–96 (2005)
18. Dong, X., Halevy, A., Yu, C.: Data integration with uncertainty. VLDB, pp. 687–698 (2007)
19. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: a survey. IEEE Trans. Knowl. Data Eng. **19**(1), 1–16 (2007)
20. Getoor, L., Diehl, C.: Link mining: a survey. SIGKDD explorations (2005)
21. Gupta, R., Sarawagi, S.: Creating probabilistic databases from information extraction models. VLDB, pp. 965–976 (2006)
22. Halevy, A., Franklin, M., Maier, D.: Principles of dataspace systems. PODS, pp. 1–9 (2006)
23. Hernández, M., Stolfo, S.: Real-world data is dirty: data cleansing and the merge/purge problem. Data Mining Knowledge Dis. **2**(1), 9–37 (1998)
24. Ioannou, E., Nejdl, W., Niederée, C., Velegrakis, Y.: On-the-fly entity-aware query processing in the presence of linkage. PVLDB **3**(1), 429–438 (2010)
25. Ioannou, E., Nejdl, W., Niederée, C., Velegrakis, Y.: LinkDB: a probabilistic linkage database system. SIGMOD conference, pp. 1307–1310 (2011)
26. Ioannou, E., Niederée, C., Nejdl, W.: Probabilistic entity linkage for heterogeneous information spaces. CAiSE, pp. 302–316 (2008)
27. Kalashnikov, D., Mehrotra, S.: Domain-independent data cleaning via analysis of entity-relationship graph. ACM Trans. Database Syst. **31**(2), 716–767 (2006)
28. Lenzerini, M.: Data integration: a theoretical perspective. PODS, pp. 233–246 (2002)
29. Morris, A., Velegrakis, Y., Bouquet, P.: Entity identification on the semantic web. SWAP (2008)
30. Papadakis, G., Ioannou, E., Niederée, C., Fankhauser, P.: Efficient entity resolution for large heterogeneous information spaces. WSDM, pp. 535–544 (2011)
31. Rastogi, V., Dalvi, N., Garofalakis, M.: Large-scale collective entity matching. PVLDB **4**(4), 208–218 (2011)
32. Re, C., Suciu, D.: Managing probabilistic data with MystiQ: the can-do, the could-do, and the can't-do. SUM, pp. 5–18 (2008)
33. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. KDD, pp. 269–278 (2002)
34. Sen, P., Deshpande, A.: Representing and querying correlated tuples in probabilistic databases. ICDE, pp. 596–605 (2007)
35. Velegrakis, Y.: On the importance of updates in information integration and data exchange systems. DBISP2P (2008)
36. Whang, S., Menestrina, D., Koutrika, G., Theobald, M., Garcia-Molina, H.: Entity resolution with iterative blocking. SIGMOD Conference, pp. 219–232 (2009)