Annales Univ. Sci. Budapest., Sect. Comp. 43 (2014) 39–55

THEORETICAL FOUNDATIONS OF ENTITY RESOLUTION MODELS

Csaba István Sidló (Budapest, Hungary)
András József Molnár (Budapest, Hungary)
Gábor Lukács (Budapest, Hungary)
András A. Benczúr (Budapest, Hungary)

Dedicated to András Benczúr on the occasion of his 70th birthday

Communicated by János Demetrovics

(Received June 1, 2014; accepted July 1, 2014)

Abstract. Data quality is crucial in all information systems. As a key step in obtaining clean data, record linkage or entity resolution (ER) groups database records by the underlying real world entities. In this paper we give practical motivating examples and review the available ER formal models. The formal model for matching and merging records determines not just the power and quality, but also the algorithmic cost of the resolution process. Starting from a naive definition that may lead to unbounded entities or infinite loops and also discussing the shortcomings of the standard axioms, we give algebraic properties that lead to efficient record partitioning. Finally we describe algorithms suitable for complex entity resolution problems that may include fuzzy clustering to split a partition of records into potentially overlapping entities.

Key words and phrases: entity resolution, deduplication, record linkage, data quality, lattice theory

The publication was supported by the grant OTKA NK 105645, the EC FET Open project "New tools and algorithms for directed network analysis" (NADINE No 288956), and the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project is supported by the European Union, co-financed by the European Social Fund.

https://doi.org/10.71352/ac.43.039

1. Introduction and related work

Data are of high quality "if they are fit for their intended uses in operations, decision making and planning" [11]. Emerging technologies for "Big Data", data science and NoSQL drive the development of advanced methods and tools for data management [17]. As a central step in obtaining quality data, entity resolution (ER) [13] is the task of identifying duplicate records and merging them into a single entity with clean attributes inherited from the original, potentially incomplete records.

In this paper, we give an overview of the available entity resolution methods, and after defining a generic formal model, we discuss the practical constraints of various approaches. We show how entity resolution in practice can be reduced to the problem of finding connected components of linked entities and then breaking up these components into possibly overlapping groups of entities.

In our previous work we gave efficient distributed algorithms for large scale entity resolution problems [19] and described theoretical restrictions to keep the methods feasible for over millions of records [14]. In this paper we concentrate on the theoretical property of entity lattices arising in different variants of the problem.

The seminal work of Fellegi and Sunter [8] introduced the problem of record linkage, which has been discussed under several different names later, e.g. duplicate detection, record matching or linkage, merge/purge or entity resolution. These names basically refer to the problem of finding and grouping together all records of real-world entities, for example all records of a client, of an item, or all Web pages referring to a person. Entity resolution is considered to be a major task of data quality assurance; [20] describes the process in this general context.

Early results (see [7] for a survey) concentrate on methods for pairwise record matching, metrics reflecting business needs and optimization methods speeding up the matching process (the join operators). A drawback of the pairwise record matching approach is that real-life data sets usually contain record pairs not similar at first, but which can be linked to the same entity in multiple inference steps. The simplest models enabling indirect inference employ partial algebras of records with an arbitrary match relation. These may however result in unbounded entities (when for example the merge of two entities concatenates their content), or infinite loops (if the merge overwrites the previous content with the new).

Some of the early results, such as [9] or [3], see the problem as a link mining task, useful for example to process Web data.

The influential Swoosh algorithms in [2] consider ER as an iterative process that finds indirect connections between entities by using black box functions

for matching and then merging entities. These black box functions should satisfy four properties, Idempotence, Commutativity, Associativity and Representativity (ICAR) [2]. Not emphasized in their original paper, ICAR is a slight weakening of an equivalence relation over entities in that associativity is required only for matching entities and in practice typically an equivalence relation is assumed.

Efficiency considerations impose additional constraints orthogonal to the ICAR properties. ER is a complex problem since finding matching input item pairs already has $O(n^2)$ time complexity over n records. Performance issues are therefore crucial, especially when dealing with large data sets. We show that if we impose additional constraints, ICAR turns to an equivalence relation, i.e. the set of records is partitioned by the merge operation, which can be computed efficiently. The record equivalence relation based entity resolution solutions are however overly restrictive and among others do not allow overlapping entities. We describe the connection between more practical non-equivalence and the more easily computable equivalence relation-based problems.

Recently, several approaches were published that provide speed and scalability. Indexing methods are surveyed in [5]. Blocking methods are described in [23] in detail. As ease in implementation while keeping flexibility, the model is defined by attributes and combination of attributes called *features* by [13]. Our previous results on using indexes for ER is described in [18]. Parallelization as a common technique to build scalable systems is also used increasingly. That is not surprising if we look at how widespread distributed platforms and NoSQL solutions became in the last decade. For pairwise matching, distributed set-similarity join methods are published, e.g. [21]. We compare the usability of three distributed software frameworks for entity resolution in [19].

The key of efficiency in all approaches is to gradually build entities by extending them with a single record. As a step beyond ICAR abstraction, we give natural algebraic requirements of accessibility and absorptivity that guarantees the correctness of the single record methods. Our main Theorem 2 states that if the two new requirements are added to (even a weakening of) ICAR, the entities arise as subsets of the records. We define the union of all overlapping entities as the *entity closure*, a partition of the original records, and show that the entity closure can be efficiently computed.

A drawback of all the efficient algorithms, which build entities as a partition of the record set, is that they cannot handle more complex tasks such as identifying overlapping households in customer records. As a natural requirement, we would like to allow entities with fuzzy boundaries and overlaps, for example by using data confidences [13] or evolving rules [22].

In our approach we consider entity resolution as a clustering problem, where many relatively small clusters have to be found that will eventually form potentially overlapping entities. Our algorithm first builds the entity closure and then uses a combination of the standard agglomerative clustering and the Apri-

Figure 1. Matching customer records.

ori algorithms [16, Sections 8.3 and 6.2]. Note, however, that by the negative results of [12], it is unlikely that the structure of overlapping entities has an algebraic characterization similar to the entity closure task.

The rest of this paper is organized as follows. Section 2 introduces use cases and practical examples as motivation. Section 3 is devoted to defining various algebraic formulations of entity resolution, including the ICAR properties [2] and our new notions of accessibility and absorptivity, with the main Theorem 2 in Section 3.3. Section 4 describes algorithms for various restricted problems, including the entity closure, the clustering approach for overlapping entities, and temporal evolving entities such as households that change in time.

2. Motivation

ER problems appear in several fields of application, for example problems of customer, product, Web, map or location data all occur in practice. We focus on customer data as a main application area, since CRM, master data management and analytical solutions all require precise knowledge of the user base. The following common practical use cases motivate the development of our ER models.

Figure 1 illustrates a common scenario with rule-based matching of personalities, where the goal is to group and merge customer records according to the real-life customer entities. Here pairwise matching is carried out based on ID or email address equality, until we get an entity consisting all four records. Note, that e.g. the third and fourth records do not match directly, we can reason only indirectly that they belong to the same person.

Figure 1 depicts records as structured relational tuples of attributes. The ER problem however can be interpreted on semi-structured, set based or arbitrary elements, which we also call records in the following.

As the output of the resolution process, we may obtain entities different from the primary type of the individual data elements. For example, we can

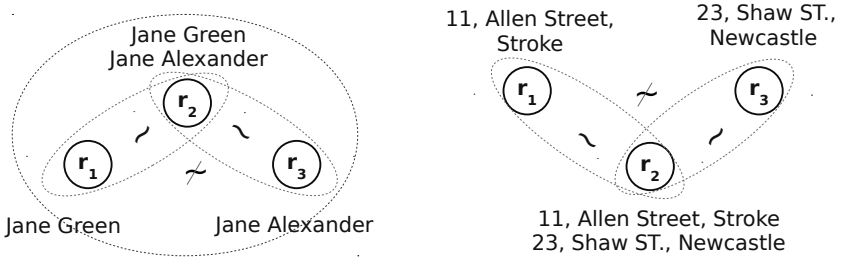

Figure 2. Overlapping household entities of personal identity records.

create households from customer records, as seen in the example of Figure 2. On the left hand side identical names suggest that records only differ because of a name change after marriage, therefore three records are grouped together in one household. On the right hand side, however, in a similar scenario, postal addresses suggest the existence of two households. A record r_2 contains both addresses (as mailing and permanent addresses for example), for a person who belongs to two overlapping but different households.

In the common case of overlapping entities, the match relation is not transitive: if r_1 is in relation with r_2 and r_2 with r_3, then r_1 is not necessarily in relation with r_3. We can also observe that entities are often time-dependent: a person and the corresponding data elements may belong to different households at different times.

Matching may use similarity and fuzzy measures, e.g. two identity records might be merged as a household entity if postal addresses and family names are similar enough. This brings further complexity to ER: as Figure 3 depicts, entities might not match until all the records of the two sides are matched.

As an example of an entity that cannot be formed from its records in an obvious way, let household entities be merged by an address-based feature. If matching is using validity date intervals (somebody lived somewhere between two dates) with a threshold on the number of days with the same address, then intermediate entities might not match until we get the final union of intervals.

Entity boundaries might be fuzzy, too. For example, somebody moving from one household to an other may result in records associating him or her to both. Somebody belonging to a household with a partner and to an other one with the parents to some extent, at the same time, also results in overlapping, not accurately defined entities. As another example, a person may appear as a private customer and at the same time the representative of a business customer. It is a question of business needs if such client entities should overlap, should be merged or should be kept distinct in such cases.

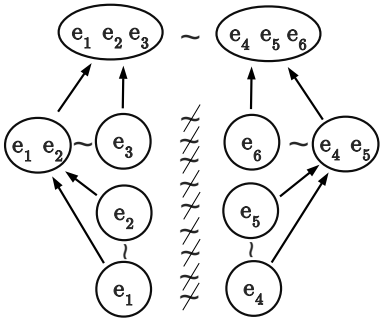

Figure 3. Entities that match only after several merge steps.

3. Defining entity resolution problems

The simplest deduplication task is finding the matching pairs of an R record set. Finding record pairs is not considered to be an ER task, but the model and methods discussed in this paper are still useful for solving it. Pairwise matches, a.k.a. similarity joins without merging, for example, can be supported by feature indexing of Section 3.7. Next we provide a more general ER model, derived from our previous model in [14].

Let a set of **records** be $R = \{r_1, r_2, \ldots, r_m\}$, and E be a set of **entities**, with $R \subseteq E$. Records represent input elements to be merged, considered to be simple entities initially. (As an alternative definition entities could be defined as sets of records, but the generic model dealing with entities proved to be more useful.) Let a **match** function be a Boolean function over $E \times E$. A corresponding **merge** is a partial function, mapping matching entity pairs to entities. A match is denoted as $\cdot \sim \cdot$, a merge as $\langle \cdot, \cdot \rangle_{\sim}$, or simply $\langle \cdot, \cdot \rangle$.

Let $\mathcal{E} = (E, \langle \cdot, \cdot \rangle)$ be a partial algebra, where entities are generated by the record set:

$$E_0 = R, \tag{3.1}$$

$$E_i = E_{i-1} \cup \{\langle x, y \rangle | x, y \in E_{i-1} \wedge x \sim y\}, \tag{3.2}$$

$$E = \bigcup_{i=0}^{\infty} E_i. \tag{3.3}$$

Let the **entity resolution** $ER(\mathcal{E})$ be the set of all maximal elements of E, i.e. the elements that cannot be extended by merging matching elements, i.e. performing the valid matches:

$$ER(\mathcal{E}) = \{x \in E \mid \forall y \in E : (y \sim x \Rightarrow \langle y, x \rangle = x) \ \wedge (x \sim y \Rightarrow \langle x, y \rangle = x)\}. \tag{3.4}$$

Our ER model is flexible, allowing E to contain sets of records, lists, semi-structured or any other data structures. E is however not necessarily finite, and also maximal elements might be generated by applying the merge operator infinitely many times. For example, a merge function concatenating record values results in values growing ever longer. As another, cyclic example, if the match function keeps the most recently added value, then E may not contain maximal elements at all.

3.1. ICAR and domination

The "generic" ER model with ICAR conditions of [2] is considered a standard definition of the entity resolution problem, and is used often in practice. The model and methods are generic in the sense that they use black box match and merge functions without studying the internal details of these functions. ICAR stands for the followings properties:

Idempotence: For all $e \in E$: $e \sim e$ and $\langle e, e\rangle = e$. An entity always matches itself, and merging it with itself still yields the same entity.

Commutativity: For all $e_1, e_2 \in E$: $e_1 \sim e_2$ iff $e_2 \sim e_1$, and if $e_1 \sim e_2$, then $\langle e_1, e_2\rangle = \langle e_2, e_1\rangle$.

(Weak) Associativity : For all $e_1, e_3, X \in E$ such that $\langle\langle e_1, X\rangle, e_3\rangle$ and $\langle e_1\langle X, e_3\rangle\rangle$ exist: $\langle\langle e_1, X\rangle, e_3\rangle = \langle e_1, \langle X, e_3\rangle\rangle$.

Representativity: If $e_1, e_2, e_3 \in E$ and $e_3 = \langle e_1, e_2\rangle$, then for any $e_4 \in E$ such that $e_1 \sim e_4$, we also have $e_3 \sim e_4$.

Representativity is a strong condition not allowing overlapping or fuzzy entities. When searching for households for example, this strict requirement causes problems.

Although the original paper [2] defines the third property as associativity, it is in fact weaker than the usual definition that also requires the existence of $\langle r_1, \langle r_2, r_3\rangle\rangle$, whenever $\langle\langle r_1, r_2\rangle, r_3\rangle$ exist. The stronger version of associativity does not hold in practice either, since r_1 may share an attribute with r_3 but may be distinct from r_2. In the example of Fig. 2, r_1 may not be similar to r_2 while $\langle\langle r_1, r_2\rangle, r_3\rangle$ may exist and correspond to the person as an entity. This type of ER is called "consistent ER" in [1]: If there exist multiple derivations involving the same set of records, then they should all produce the same result.

In addition to ICAR, entity domination as a partial ordering on entities is used to capture maximal elements: An e_1 entity dominates e_2 if $e_1 \sim e_2$, and $\langle e_1, e_2\rangle = e_1$. The entity resolution of [2] is then defined as the derived entity set not containing dominated elements, where domination corresponds to the maximality property of (3.4).

3.2. Accessibility and absorptivity

We reduce the complexity of the ER problem but weaken ICAR by introducing the following constraints, holding for all $e_1, e_2, e_3 \in E$ where the appropriate merge exists, i.e. where the merge operands match:

$$\textbf{idempotence}: \langle e_1, e_1 \rangle = e_1, \tag{3.5}$$

$$\textbf{commutativity}: \langle e_1, e_2 \rangle = \langle e_2, e_1 \rangle, \tag{3.6}$$

$$\textbf{associativity}: \langle \langle e_1, e_2 \rangle, e_3 \rangle = \langle e_1, \langle e_2, e_3 \rangle \rangle. \tag{3.7}$$

For $e \in E$, let $rank(e)$ be the minimum number of generator elements in R, with multiplicities, needed to generate e. Similar to [4], let the **accessibility** property be the following:

$$\begin{aligned} &\text{For all } e \in E \text{ there is an } r \in R \\ &\text{and an } x \in E \text{ with } rank(x) < rank(e) \\ &\text{such that } e = \langle x, r \rangle. \end{aligned} \tag{3.8}$$

Let an accessible entity be $e = \langle \ldots \langle \langle r_1, r_2 \rangle, r_3 \rangle \ldots, r_k \rangle$, where $k > 1$. The operators are **absorptive**, if for all $j \in 1..k$:

$$\begin{aligned} e \sim r_j &\Rightarrow \langle e, r_j \rangle = e, \\ r_j \sim e &\Rightarrow \langle r_j, e \rangle = e. \end{aligned} \tag{3.9}$$

3.3. The accessible subset: Equivalence with set union

Having the requirements (3.5) through (3.9), E with the merge operation is a partial semilattice of records. We show that this semilattice has an isomorphic partial semilattice with equivalence classes of 2^R (subsets of R) and a partial set union operator. As a consequence, we may simply consider subsets of R instead of general expressions generated by R. The accessible subset model then uses the properties of Section 3.2 and the set union operator.

Theorem 1. *Let S_E be the set of record subsets in which the records have at least one valid sequence of merge operations. There exists a homomorphism from record subsets $S_E \subseteq 2^R$ with a partial set union operator $\cup_\sim$ to the set of entities E with an idempotent, commutative, associative, accessible and absorptive merge operator.*

Proof. Let the relation $\cup_\sim$ be the set union operator for sets in S_E. Let the mapping $\phi : S_E \to E$ be the following, for all $r \in R$:

$$\begin{aligned} \phi(\{r\}) &= r, \\ \phi(A \cup_\sim \{r\}) &= \langle \phi(A), r \rangle \text{ for all } A \in S_E. \end{aligned}$$

We show that ϕ is well defined, i.e. there is no ambiguity for any set B in the recursive definition of a $\phi(B)$, regardless of how an r and A is chosen with $A = B \setminus \{r\}, B = \{r\} \cup A$, and preserves the operation. That is, for all $A, B \in S_{\scriptscriptstyle E}$: $\phi(A \cup_{\sim} B) = \langle \phi(A), \phi(B) \rangle$.

For sets containing only one record ϕ is well defined. It preserves the operation, because merge is idempotent: $\forall r \in R : r \sim r, \langle r, r \rangle = r$. Next we consider $\{r_1, r_2, r_3 \ldots, r_n\} \in S_{\scriptscriptstyle E}$, with $n \in \mathbb{N}$ and $i \in [1, n], r_i \in R$.

First we show that the record merge order has no effect on the merge result. The merge sequence does not necessarily exist for all the merge sequence variants, because merge is only defined for matching entities, but if the merge exists, associativity and commutativity ensures that different record merge orderings result in the same entity.

Second, we show that all elements $x \in E$ can be constructed by merging records $r \in R$ in a way that all records are used only once. If a record $r \in R$ occurs in the merge sequence of an entity $x \in E$ multiple times, it has no effect on the merged element. Because the order has no effect on the result and absorptivity holds, we only have to show that every $x \in E$ can be generated as

$$\langle \ldots \langle \langle r_1, r_2 \rangle, r_3 \rangle \ldots, r_k \rangle$$

using every generator r_i at most once. Let us take the lowest rank counterexample x and use accessibility to get $x = \langle y, r \rangle$. If r appears in the description of y using every generator at most once, then $x = y$ by absorption and hence x is not a counterexample. Otherwise $\langle y, r \rangle$ is the required form of x.

Order-independence and multiplicity-independence ensures that ϕ is well-defined and the merge operation behaves as the set union. The definition of ϕ ensures operation-preservation for $\phi(A \cup B)$ where $|B| = 1$. For larger sets, operation preservation can be proven by induction.

The accessibility property of the merge and the construction $\phi(A \cup_{\sim} \{r\}) = \langle \phi(A), r \rangle$ of ϕ ensures that $\langle \phi(A), \phi(B) \rangle$ can be constructed by merging the records of A and B. Because ϕ is well-defined and the merge is order- and multiplicity-independent, $\langle \phi(A), \phi(B) \rangle = \phi(A \cup_{\sim} B)$. ■

Given the mapping ϕ, entities in E could have multiple representatives in $S_{\scriptscriptstyle E}$. In order to define a union operator over the set of entities, we show that entities of E can be substituted by elements in equivalence classes of $S_{\scriptscriptstyle E}$. By the next theorem, the merge operation behaves the same way as the set union operator.

Theorem 2. *A set of entities E with an idempotent, commutative, associative, accessible and absorptive merge operation is isomorphic to equivalence classes of $S_{\scriptscriptstyle E} \subseteq 2^R$ with the partial set union operator $\cup_{\sim}$.*

Proof. Let sets $A, B \in S_{\scriptscriptstyle E}$ be equivalent if $\phi(A) = \phi(B)$ using the homomorphism ϕ of Theorem 1. By the construction of ϕ, the equivalence classes are

well defined. The proof follows by mapping the entities in E to the equivalence classes of S_E. ■

The original $(E, \langle\cdot,\cdot\rangle)$ problem can be solved on S_E with the partial set union operator. To decide whether the union of two elements in S_E exists (if they match), the corresponding elements in E have to be found and matched. That is, a merge sequence of the records have to be constructed, where all merges exist, i.e. all merge operands match.

3.4. The entity closure

We introduce further constraints to facilitate practical use: a match function which is an equivalence relation leads us to the entity closure problem.

Let the **entity closure** for an $ER(S_E, \cup_\sim)$ entity resolution problem be $ER(S_E, \cup_{\sim^*})$, where $\sim^*$ is the transitive, symmetric and reflexive closure of the $\sim$ match function.

3.5. Relational records

Optionally, we may define the structure of records similar to unnamed relational algebra tuples. For a fixed k and for each $i \in [1..k]$, let the ith **attribute** be a function $a_i : R \to DOM_i^* = DOM_i \cup \{\emptyset\}$, where DOM_i can be any set as the attribute domain, and $\emptyset$ denotes missing values (NULL). This way an r **relational record** is described by k number of **attribute values** $a_1(r)$, ..., $a_k(r)$.

3.6. Features

Features facilitate expressing match relations, in a way that suits real-world business concepts and ER algorithms well. Three distinguishing aspects are used to describe domain knowledge of entity matching. First, independent properties of the entities are isolated as feature functions. Second, the entity match relation is transformed using these properties. Third, efficient computation is supported by assigning representative values to feature values.

Let a **feature-based match function** of a feature f be a Boolean partial function on $F \times F$.

A feature-based matching of entities through the feature function is denoted as $\sim_f$. To combine features $f_1, \ldots, f_k$, let the match of entities $e, e' \in E$ be

$$e \sim_{f_1,\ldots,f_k} e' \Leftrightarrow e \sim_{f_1} e' \vee e \sim_{f_2} e' \vee \ldots \vee e \sim_{f_k} e'.$$

3.7. Feature indexes

Indexable features support fast evaluation of feature-based matches by enabling the use of usual indexing techniques. Let the **feature mapping function** of a feature f be a partial function $rep_f : F \to 2^O$ with some ordered set O, assigning representative elements to feature values. A feature f is **indexable**, if a feature mapping function exists so that all matching entities $e, e' \in E$ can be found through the equality of representative values

$$e \sim_f e' \Rightarrow \exists\ o \in O : o \in rep_f(f(e)) \wedge o \in rep_f(f(e')).$$

4. Algorithms for entity resolution

A naive solution to solve the ER problem would be to iterate through the input entity set and find matching pairs, as long as such pairs exist. Such algorithms, including G-swoosh [2, Algorithm 2] and even the improved R-swoosh, run in quadratic time and are hence inefficient for large data sets.

The starting point of our results is F-swoosh [2, Algorithm 4] that merges records by compositions of matching attribute values. In that algorithm, the graph of entity mergers along matching values of various attributes may form an arbitrary graph. The need for connected component identification is described first in the iterative blocking algorithm of [23]. Highly efficient distributed implementations are given in [19].

4.1. Solving the entity closure problem

As a first task, we give an algorithm for computing the entity closure as defined in Section 3.4. Finding the entity closure is equivalent to the problem of finding maximal connected components of the graph over records with edges corresponding to matching records.

Algorithm 1 works by iterating through all the features and all the record pairs to enumerate the edges in the record graph. The last step of Algorithm 1 calls a connected component algorithm in the graph formed by the match relation. Feature indexes are used to narrow down the amount of record pairs to be checked for matching, by providing match candidates. If two records match according to the given feature, then they will be match candidates.

Algorithm 1 Feature-based Entity Closure

input: Record set R of entities
output: $ER(S_E, \cup_\sim)$ generated by R, where $S_E \subseteq 2^R$

```
1: for all features f_i do
2:   for all representative values rep_i(r) of r ∈ R do
3:     for all pairs of records r, r' with rep_i(r) = rep_i(r') do
4:       if r ∼_i r' then
5:         add (r, r') edge to graph G
6: find and output connected components of G
```

4.2. Clustering non-transitive entities

Although several methods such as R-swoosh [2] solve the Entity Closure problem, its limitation is that it does not allow fuzzy matching by confidence levels or learning based approaches. In fact, in most of the previous work [15, 6, 10] an exact match function is assumed, where a Boolean pairwise match function is used, with no fuzzy merge and confidence values.

We progress beyond Entity Closure limitations by considering the closures as efficient candidates. We consider the entity closure output as an efficient way to distribute the work of sophisticated match functions. Note that splitting by post-processing is proposed among others in [23]. Also, we may easily accommodate complex match criteria or even use our solutions to distribute machine learning.

The entity closure is an upper bound for the original entity resolution: the solution of the original problem can be reached by separating closure entities. Real-world entities are usually small, with minor number of instances where transitivity, commutativity or idempotence is violated. Therefore, we expect the computation of the closure to be profitable, despite of the necessary post-processing steps splitting the entities.

Standard agglomerative clustering algorithms [16, Section 8.3] are capable of breaking the elements of the entity closure into smaller entities, resulting in non-overlapping entities. For all pairs of records, we may define weight or confidence values that the pair of records match. Entity closure uses the single link strategy, i.e. if two entities have two elements with weight above a minimum threshold, the corresponding entities will be merged. Given the entity closure, we may efficiently implement any other strategies (complete link, average weight, etc.) to partition the entity closure.

In order to obtain overlapping entities, we give an apriori-like [16, Section 6.2] level-wise algorithm next. In Algorithm 2 we start out from all records as single-element entity candidates. In iteration k, we build $(k+1)$-element candidates $\mathcal{C}_k$ from k-element ones by testing the match likelihood for all ele-

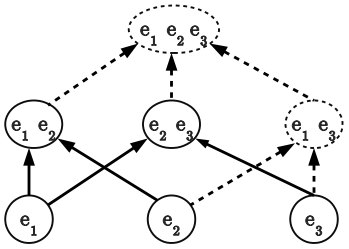

Figure 4. Simple example of a semilattice and a partial semilattice.

ments. For match likelihood computation, we may use any of the agglomerative clustering strategies (single, complete linkage, etc.). If there is no matching element, the candidate forms an entity. Otherwise, new candidates $\mathcal{C}_{k+1}$ are generated. The collection of $\mathcal{C}_k$ for all k form a semilattice as illustrated in Fig. 4.

Algorithm 2 Level-wise algorithm for finding overlapping entities

```
input: a set C of the entity closure
C_1 ← {r : r ∈ C}
for k = 1, ..., |C| do
  for all X ∈ C_k do
    entity ← true
    for all r ∈ C − X do
      if r ∼ X then
        C_{k+1} ← C_k ∪ {X ∪ {r}}
        entity ← false
    if entity = true then
      add X to the output entities
```

By the negative results of [12], it is unlikely that the structure of overlapping entities has an algebraic characterization similar to the entity closure task.

4.3. Temporal entities

Entity boundaries, as in the motivating example of household resolution, might be fuzzy. A person may for example move from the parents and therefore belong to multiple households. Defining the model with overlapping entities results in a more complex problem however. We show a simple process for time-dependent entities to reduce the problem to the entity closure, avoiding the complexity overhead of computing overlapping entities.

Our main assumption is that records in practice are often snapshots of the entity's state at a particular point in time. They also frequently employ

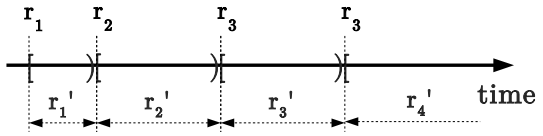

Figure 5. Snapshot- and time-interval records of an entity.

attributes indicating the time of recording. From time to time, we get the up-to-date properties of the entity, and have no updates in between these points of time. Figure 5 depicts how records of an entity are recorded, and how we can interpret them on a timeline. We assume the entity to be unchanged between the observations $(r_1, r_2, \ldots)$, and transform the records to describe states for validity time intervals of the entity $(r'_1, r'_2, \ldots)$.

Match features can be extended to match records only if they have feature values with overlapping time intervals. The size of the intersecting time interval can also be used as a weight to decide matches. For example, we may consider two identity records belonging to the same household if they share a postal address for a sufficiently long time.

These new temporal records with the extended match logic often fit the practical use cases better than the original record set. Moving between households for example can be captured more easily: Resulting entities may consist of records indicating that someone lived in a household between two exact time points instead of the original broader statement.

For another example, when resolving identity records to find people, their roles might change in time. Someone might be a client for some time as an individual, and later on be a representative of a company as a legal entity. They are usually considered as different entities, which are however hard to be differentiated without using temporal records.

Note, that such a transformation is not always possible: if, for example, somebody belongs to a household with a partner and to an other one with the parents to some extent, at the same time.

Transforming snapshot records to temporal records can be achieved by sorting the snapshot records according to the timestamp and forming the time intervals using the adjacent records.

5. Conclusions and future work

In this paper we introduced a generalized and flexible theoretical approach for entity resolution (ER) problems, including a formal framework to define matching. We described practical distributed ER algorithms and demonstrated

the usability of our methods by identifying clients and households of insurance client records. We showed that even complex ER tasks involving overlapping entities, for example households, can be efficiently solved by first computing the entity closure and then splitting the components.

As an open theoretical problem, we pose the question of finding algebraic definitions of entity resolution problems that result in entity structures more complex than the equivalence relation in Theorem 1 or the entity closure of Section 3.4.

References

[1] **Benjelloun, O., H. Garcia-Molina, H. Kawai., T.E. Larson, D. Menestrina, Q. Su, S. Thavisomboon and J. Widom,** *Generic entity resolution in the serf project,* Tech. report 2006-14, Stanford InfoLab, 2006.

[2] **Benjelloun, O., D. Garcia-Molina, D. Menestrina, Q. Su, S.E. Wang and J. Widom,** Swoosh: A generic approach to entity resolution, *VLDB J.,* **18** (1) (2009), 255-276.

[3] **Bhattacharya, I. and L. Getoor,** Collective entity resolution in relational data, *ACM Trans. Knowl. Discov. Data,* **1** (1:5) (2005).

[4] **Boley, M., T. Horváth, A Poigné and S. Wrobel,** Efficient closed pattern mining in strongly accessible set systems, *Knowledge Discovery in Databases: PKDD 2007,* 382-389.

[5] **Christen, P.,** A survey of indexing techniques for scalable record linage and deduplication, *IEEE Trans. on Knowledge and Data Engineering,* **99** (PrePrints), 2011.

[6] **Dong, X., A. Halevy and J. Madhavan,** Reference reconciliation in complex information spaces, *Proc. 2005 Int. Conf. on Management of Data,* ACM, 2005, 85-96.

[7] **Elmagarmid, A., P. Ipeirotis and V. Verykios,** Duplicate record detection: A survey, *IEEE Trans. on Knowledge and Data Engineering,* (2007), 1-16.

[8] **Fellegi, I. and A. Sunter,** A theory for record linkage, *J. Amer. Stat. Assoc.,* **64** (328) (1969), 1183-1969.

[9] **Getoor, L. and C. Diehl,** Link mining: A survey, *ACM SIGKDD Explorations Newsletter,* **7** (2) (2005), 3-12.

[10] **Hernández, M. and S. Stolfo,** Real-world data is dirty: Data cleaning and the merge/purge problem, *Data Mining and Knowledge Discovery,* **2** (1) (1998), 9-37.

[11] **Juran, J. and A.B. Godfrey,** *Quality Handbook,* McGraw-Hill, 1999.

[12] **Kleinberg, J.M.,** An impossibility theorem for clustering, *NIPS,* (2002), 446-453.

[13] **Menestrina, D., O. Benjelloun and H. Garcia-Molina,** Generic entity resolution with data confidences, *CleanDB Workshop, 2006,* 25-32.

[14] **Molnár, A.J., A.A. Benczúr and C.I. Sidló,** Flexible and efficient distributed resolution of large entities, *Proc. 7th Int. Conf. on Foundations of Information and Knowledge Systems FoIKS'12,* Springer, 244-263.

[15] **Monge, A. and C. Elkan,** An efficient domain-independent algorithm for detecting duplicate database records, *SIGMOD DMKD,* 1997.

[16] **Pang-Ning, T., M. Steinbach, V. Kumar et al.,** *Introduction to data mining,* WP Co, 2006.

[17] **Roebuck, K.,** *Data Quality: High-impact Strategies - What You Need TO Know: definitions, Adoptions, Impact, Benefits, Maturity, Vendors,* Emereo Pty Lim., 2011.

[18] **Sidló, C.I.,** Entity resolution with heavy indexing, *Proc. 2011 Int. Conf. on Advances in Databases and Information Systems,* CEUR Workshop Proceedings, 2011.

[19] **Sidló, C.I., A. Garzó, A. Molnár and A.A. Benczúr,** Infrastructures and bounds for distributed entity resolution, *9th Int. Workshop on Quality in Databases in conjuction with VLDB 2011,* 2011.

[20] **Talburt, J.,** *Entity Resolution and Information Quality,* Elsevier Science, 2011.

[21] **Vernica, R., M. Carey and C. Li,** Efficient parallel set-similarity joins using mapreduce, *Proc. 2010 Int. Conf. on Management of Data,* ACM, 2010, 495-506.

[22] **Whang, S.E. and H. Garcia-Molina,** Entity resolution with evolving rules, *Proc. VLDB Endow.,* **3** (2010), 1326-1337.

[23] **Whang, S.E., D. Menestrina, G. Koutrika, M. Theobald and H. Garcia-Molina,** Entity resolution with iterative blocking, *Proc. 35th Int. Conf., on Management of Data,* ACM, 2009, 219-232.

Csaba István Sidló
Gábor Lukács
András A. Benczúr
Computer and Automation Institute
Hungarian Academy of Sciences
(MTA SZTAKI)
University of Debrecen
Eötvös Loránd University
Budapest, Hungary
{sidlo,lukacsg,benczur}@ilab.sztaki.hu

András József Molnár
Research Institute for National Strategy
NSKI
Budapest, Hungary
jozsef.andras.molnar@nski.gov.hu