General Context-Aware Data Matching and Merging Framework

Slavko ŽITNIK[1], Lovro ŠUBELJ[1], Dejan LAVBIČ[1],
Olegas VASILECAS[2], Marko BAJEC[1]

[1]*University of Ljubljana, Faculty of Computer and Information Science*
 Tržaska cesta 25, SI-1000 Ljubljana
[2]*Information Systems Research Laboratory, Vilnius Gediminas Technical University*
 Saulėtekio 11, LT-10223 Vilnius, Lithuania
e-mail: {slavko.zitnik, lovro.subelj, dejan.lavbic}@fri.uni-lj.si, marko.bajec@fri.uni-lj.si,
 olegas@fm.vgtu.lt

Abstract. Due to numerous public information sources and services, many methods to combine heterogeneous data were proposed recently. However, general end-to-end solutions are still rare, especially systems taking into account different context dimensions. Therefore, the techniques often prove insufficient or are limited to a certain domain. In this paper we briefly review and rigorously evaluate a general framework for data matching and merging. The framework employs collective entity resolution and redundancy elimination using three dimensions of context types. In order to achieve domain independent results, data is enriched with semantics and trust. However, the main contribution of the paper is evaluation on five public domain-incompatible datasets. Furthermore, we introduce additional attribute, relationship, semantic and trust metrics, which allow complete framework management. Besides overall results improvement within the framework, metrics could be of independent interest.

Keywords: entity resolution, redundancy elimination, semantic elevation, trust, ontologies.

1. Introduction

Heterogeneous data matching and merging is due to increasing amount of linked and open (on-line) data sources rapidly becoming a common need in various fields. Different scenarios demand for analyzing heterogeneous datasets collectively, enriching data with some on-line data source or reducing redundancy among datasets by merging them into one. Literature provides several state-of-the-art approaches for matching and merging, although there is a lack of general solutions combining different dimensions arising during the matching and merging execution. We propose and evaluate a general and complete solution that allows a joint control over these dimensions.

Data sources commonly include not only network data, but also data with semantics. Thus a state-of-the-art solution should employ semantically elevated algorithms (i.e., algorithms that can process data with semantics according to an ontology), to fully exploit

the data at hand. However, due to a vast diversity of data sources, an adequate data architecture also employed. In particular, the architecture should support all types and formats of data, and provide appropriate data for each algorithm. As algorithms favor different representations and levels of semantics behind the data, architecture should be structured appropriately.

Due to different origin of (heterogeneous) data sources, the trustworthiness (or accuracy) of their data can often be questionable. Specially, when many such datasets are merged, the results are likely to be inexact. A common approach for dealing with data sources that provide untrustworthy or conflicting statements, is the use of trust management systems and techniques. Thus matching and merging should be advanced to a trust-aware level, to jointly optimize trustworthiness of data and accuracy of matching or merging. Such collective optimization can significantly improve over other approaches.

The paper proposes and demonstrates a general framework for matching and merging execution. An adequate data architecture enables either pure related data, in the form of networks, or data with semantics, in the form of ontologies. Different datasets are merged using collective entity resolution and redundancy elimination algorithms, enhanced with trust management techniques. Algorithms are managed through the use of different contexts that characterize each particular execution, and can be used to jointly control various dimensions of variability of matching and merging execution. Proposed framework is also fully implemented and evaluated against real-world datasets.

The rest of the paper is structured as follows. The following section gives a brief overview of the related work, focusing mainly on trust-aware matching and merging. Next, Section 3, presents employed data architecture and discusses semantic elevation of the proposition. Section 4 formalizes the notion of trust and introduces the proposed trust management techniques. General framework, and accompanying algorithms, for matching and merging are presented in Section 5. Experimental demonstration of the proposed framework is shown in Section 6, and further discussed in Section 7. Section 8 concludes the paper.

2. Related Work

Recent literature proposes several state-of-the-art solutions for matching and merging data sources. Analogous problems appear in many different areas. When observing the area of matching and merging on a broader basis, we used ideas from different approaches in the fields of data integration (Bhattacharya and Getoor, 2004; Cohen, 2000; Hernandez and Stolfo, 1995; Lenzerini, 2002), data deduplication (Ananthakrishna *et al.*, 2002; Kalashnikov and Mehrotra, 2006; Monge and Elkan, 1996), information retrieval, schema and ontology matching (Castano *et al.*, 2006; Castano *et al.*, 2010; Euzenat and Shvaiko, 2007; Rahm and Bernstein, 2001), and (related) entity resolution (Bhattacharya and Getoor, 2004; Bhattacharya and Getoor, 2007; Dong *et al.*, 2005).

The propositions mainly address only selected issues of more general matching and merging problem. In particular, approaches only partially support the variability of the

execution, commonly only homogeneous sources, with predefined level of semantics, are employed, or the approaches discard the trustworthiness of data and sources of origin. A Mapping-based Object Matching – MOMA System (Thor and Rahm, 2007) presents the use of workflows and combination of several matching algorithms within a single data source. Our approach uses attribute resolution technique to align arbitrary data sources and prepares them for further matching and merging techniques. The general problem of many approaches over large-scale datasets is response time to first possible results. Pay-As-You-Go ER (Whang *et al.*, 2010) system maximizes entity resolution progress with a limited amount of work according to defined constraints. It orders merging pairs using these constraints and outputs partial results as soon as po ssible. We run our algorithms on network data and merge pairs according to similarity value using contexts, where the user can observe the whole network during merging and matching execution. Networks seemed the most appropriate to design our approach. They enable us to dynamically change and read structure as it is done by techniques of label propagation (Šubelj and Bajec, 2011a) or community detection (Šubelj and Bajec, 2011b) where each community presents matched data.

The proposed matching and merging approach employs the use of contexts using semantics, trust and ontologies. The problem of matching references to an underlying entity in natural language processing is known as coreference resolution (Ng, 2008). Traditionally the problem was solved using a set of constraints of features, but improvements were achieved by using multiple matching models and propagation of shared attributes across references (Lee *et al.*, 2011). The idea of using different attribute, related and semantic metrics is used from similar categorization of features on simple pairwise approach which outperformed complex coreference resolution models (Bengston and Roth, 2008). Use of ontologies, axioms and their inference as also used in text mining (Štajner and Mladenič, 2009), additionally gives us schema, knowledge modelling and control mechanism (Lavbič *et al.*, 2010) during matching and merging execution.

Literature also provides various trust-based, or trust-aware, approaches for matching and merging (Nagy *et al.*, 2008; Richardson *et al.*, 2003). Although they formally exploit trust in the data, they do not represent a general or complete solution. Mainly, they explore the idea of *Web of Trust*, to model trust or belief in different entities. Related work on *Web of Trust* exists in the fields of identity verification (Blaze *et al.*, 1996), information retrieval (Chakrabarti *et al.*, 1998; Joachims, 1997), social network analysis (Domingos and Richardson, 2001; Kleinberg, 1999), data mining and pattern recognition (Kautz *et al.*, 1997; Resnick *et al.*, 1994). Our work also relates to more general research of trust management and techniques that provide formal means for computing with trust, e.g., (Trcek, 2009). Some research has also been done on using the strategy of disinformation (Whang and Garcia-Molina, 2011). The strategy focuses on matching and merging the records with bogus information and is useful for robustness evaluation. The use of trust management context in ou r approach is defined on levels from whole source to attribute values.

This paper superseds our previously published theoretical concepts of the same system (Šubelj *et al.*, 2011). We did some minor changes to definitions of ontology usage,

renamed some notions (e.g., due to disambiguation we are referring to relations as related data) and introduced an optimization by checking only neighbouring data (19th line of Algoritm 1). The main contributions over the previous paper are experiments (Section 6) of all proposed methods and metrics on real datasets. Implementations of general components are in-depth presented and therefore it is shown the usage of semantics and trust improves overall results.

3. Data Architecture

An adequate data architecture is of vital importance for efficient matching and merging. Key issues arising are as follows: (1) architecture should allow for data from heterogeneous sources, commonly in various formats; (2) semantical component of data should be addressed properly; and (3) architecture should also deal with (partially) missing and uncertain data.

To achieve superior performance, we propose a three level architecture (Fig. 3). Standard network data representation on the bottom level (*data level*) is enriched with semantics (*semantic level*) and thus elevated towards the topmost real world level (*abstract level*). Datasets on data level are represented with networks, when the semantics are employed through the use of ontologies.

Every dataset is (preferably) represented on data and semantic level. Although both describe the same set of entities on abstract level, the representation on each level is independent from others. This separation resides from the fact that different algorithms of matching and merging execution privilege different representations of data – either pure related or semantically elevated representation. Separation thus results in more accurate and efficient matching and merging, moreover, representations can complement each other in order to boost the performance.

The following section gives a brief introduction to networks, used for data level representation. Section 3.2 describes ontologies and semantic elevation of data level (i.e., semantic level). Proposed data architecture is formalized and further discussed in Section 3.3.

3.1. *Representation with Networks*

Most natural representation of any related data are *networks* (Newman, 2010). They are based upon mathematical objects called *graphs*. Informally speaking, graph consists of a collection of points, called *vertices*, and links between these points, called *edges* (Fig. 1). Let V_N, E_N be a set of vertices, edges for some graph N respectively. We define graph N as $N = (V_N, E_N)$ where

$$V_N = \{v_1, v_2, \ldots, v_n\}, \tag{1}$$
$$E_N \subseteq \big\{\{v_i, v_j\}\,|\, v_i, v_j \in V_N \wedge i < j\big\}. \tag{2}$$
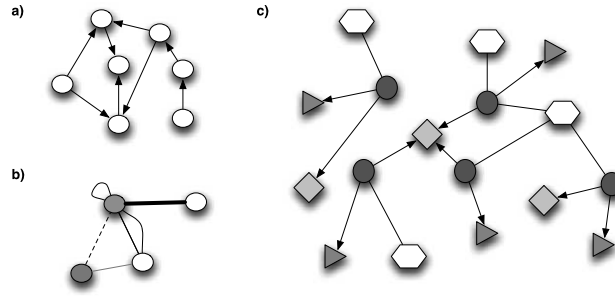
Fig. 1. (a) Directed graph; (b) labeled undirected multigraph (labels are represented graphically); (c) network representing a group of related restaurants (circles correspond to restaurants, hexagons to their types, triangles to different phone numbers, while squares represent respective cities).

Edges are sets of vertices, hence they are not directed (*undirected graph*). In the case of *directed graphs* (2) rewrites to

$$E_N \subseteq \big\{ (v_i, v_j) | \; v_i, v_j \in V_N \wedge i \neq j \big\}, \tag{3}$$

where (v_i, v_j) is an edge from v_i to v_j. The definition can be further generalized by allowing multiple edges between two vertices and *loops* (edges that connect vertices with themselves). Such graphs are called *multigraphs* (Fig. 1(b)).

In practical applications we commonly strive to store some additional information along with the vertices and edges. Formally, we define *labels* or *weights* for each node and edge in the graph – they represent a set of properties that can also be described using two attribute functions

$$A_{V_N} \colon V_N \to \Sigma_1^{V_N} \times \Sigma_2^{V_N} \times \cdots, \tag{4}$$
$$A_{E_N} \colon E_N \to \Sigma_1^{E_N} \times \Sigma_2^{E_N} \times \cdots, \tag{5}$$

$A_N = (A_{V_N}, A_{E_N})$, where $\Sigma_i^{V_G}, \Sigma_i^{E_G}$ are sets of all possible vertex, edge attribute values respectively.

Networks are most commonly seen as labeled, or weighted, multigraphs with both directed and undirected edges (Fig. 1(c)). Vertices of a network represent some entities, and edges represent related data between them. A (related) dataset, represented with a network on the data level, is thus defined as (N, A_N).

3.2. *Semantic Elevation Using Ontologies*

Ontologies are formal definitions of classes, related data, functions and other objects. An *ontology* is an explicit specification of conceptualization (Gruber, 1993), which is is an abstract view of the knowledge we wish to represent. It can be defined as a network of *entities*, restricted and annotated with a set of *axioms*. Let E_O, A_O be the sets of entities,
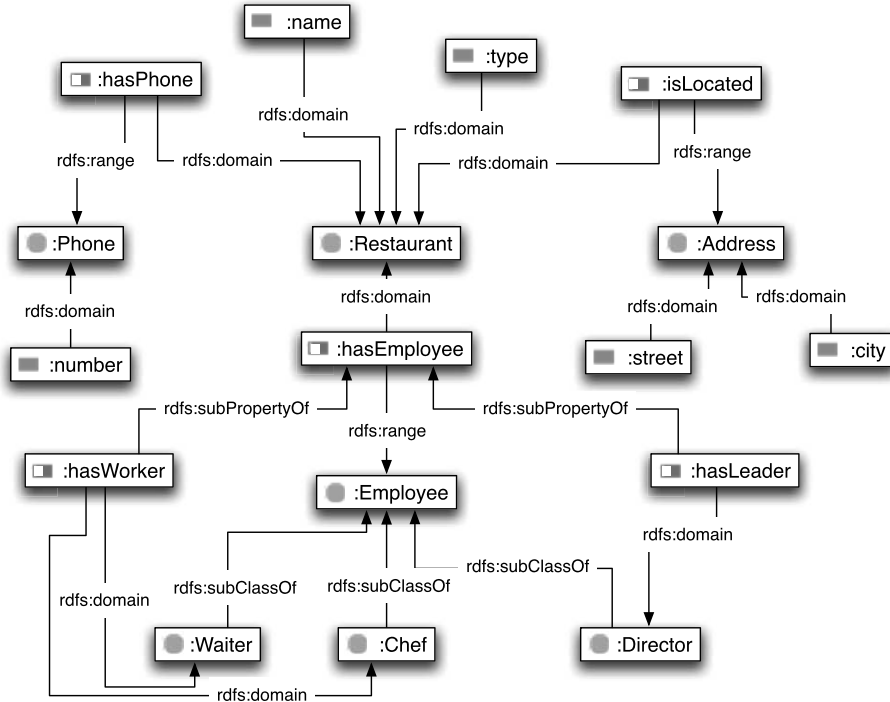
Fig. 2. A possible ontology over Restaurants dataset (description in Section 6.1). Classes are represented by circles, related data by half-white rectangles and attributes by full-colour rectangles. Key concepts of the ontology are *Restaurant*, *Address*, *Phone* and *Employee*.

axioms for some ontology O respectively. We propose a dataset representation with an ontology on semantic level (an example in Fig. 2) as $O = (E_O, A_O)$ where

$$E_O \subseteq E^C \cup E^I \cup E^R \cup E^A, \tag{6}$$

$$A_O \subseteq \left\{ a \mid E_O^a \subseteq E_O \land a \text{ axiom on } E_O^a \right\}. \tag{7}$$

Entities E_O consist of *classes* E^C (concepts), *individuals* E^I (instances), *related data* E^R (among classes and individuals) and *attributes* E^A (properties of classes); and axioms A_O are assertions (over entities) in a logical form that together comprise the overall theory described by ontology O.

This paper focuses on ontologies based on *descriptive logic* that, besides assigning meaning to axioms, enable also reasoning capabilities (Horrocks and Sattler, 2001). The latter can be used to compute consequences of the previously made assumptions (queries), or to discover non-intended consequences and inconsistencies within the ontology.

One of the most prominent applications of ontologies is in the domain of semantic interoperability (among heterogeneous software systems). While pure semantics concerns

the study of meanings, we define *semantic elevation* as a process to achieve semantic interoperability which be considered as a subset of information integration.

Thus one of the key aspects of semantic elevation is to derive a common representation of classes, individuals, related data and attributes within some ontology. We employ a concept of *knowledge chunks* (Castano *et al.*, 2010), where each entity is represented with its *name*, set of semantically *related data*, attributes and *identifiers*. All of the data about a certain entity is thus transformed into *attribute-value* format, with an identifier of the data source of origin appended to each value. Exact description of the transformation between networked data and knowledge chunks is not given, although it is very similar to the definition of inferred axioms in (12), Section 3.3. Knowledge chunks, denoted $k \in K$, thus provide a (common) synthetic representation of an ontology that is used during the matching and merging execution. For more details on knowledge chunks, and their construction from a *RDF(S)* (*Resou rce Description Framework Schema*) repository or an *OWL* (*Web Ontology Language*); see Castano *et al.* (2009, 2010).

3.3. *Three Level Architecture*

As previously stated, every dataset is (independently) represented on three levels – *data*, *semantic* and *abstract level* (Fig. 3). Bottommost *data level* holds data in a pure related format (i.e., networks), mainly to facilitate state-of-the-art related data algorithms for matching. Next level, *semantic level*, enriches data with semantics (i.e., ontologies), to further enhance matching and to promote semantic merging execution. Data on both levels represent entities of topmost *abstract level*, which serves merely as an abstract (artificial) representation of all the entities, used during matching and merging execution.

The information captured by data level is a subset of that of semantic level. Similarly, the information captured by semantic level is a subset of that of abstract level. This *information-based* view of the architecture is seen in Fig. 3(a). However, representation on each level is completely independent from the others, due to absolute separation of data. This provides an alternative *data-based* view, seen in Fig. 3(b).
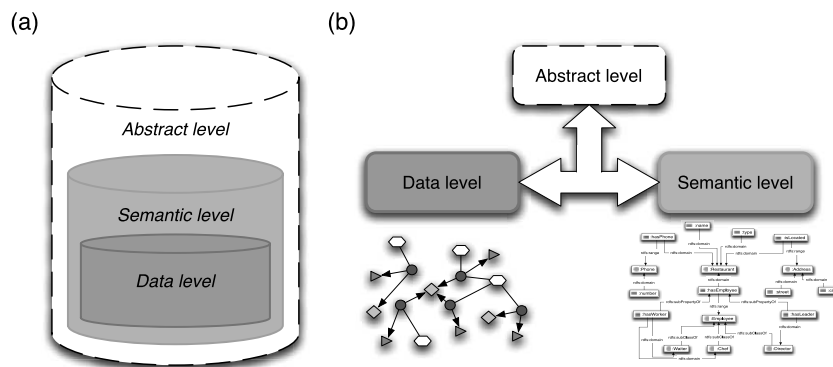

Fig. 3. (a) Information-based view of the data architecture; (b) data-based view of the data architecture.

To manage data and semantic level independently (or jointly), a mapping between the levels is required. In practice, data source could provide datasets on both, data and semantic level. The mapping is in that case trivial (i.e., given). However, more commonly, data source would only provide datasets on one of the levels, and the other has to be *inferred*.

Let (N, A_N) be a dataset, represented as a network on data level. Without loss for generality, we assume that N is an undirected network. Inferred ontology $(\tilde{E}_{\tilde{O}}, \tilde{A}_{\tilde{O}})$ on semantic level is defined with

$$\tilde{E}_C = \{\text{vertex}, \text{edge}\}, \tag{8}$$
$$\tilde{E}_I = V_N \cup E_N, \tag{9}$$
$$\tilde{E}_R = \{\text{isOf}, \text{isIn}\}, \tag{10}$$
$$\tilde{E}_A = \{A_{V_N}, A_{E_N}\}, \tag{11}$$

and

$$\begin{aligned}
\tilde{A}_{\tilde{O}} = \ &\big\{v \text{ isOf vertex} \mid v \in V_N\big\} \\
&\cup \big\{e \text{ isOf edge} \mid e \in E_N\big\} \\
&\cup \big\{v \text{ isIn } e \mid v \in V_N \wedge e \in E_N \wedge v \in e\big\} \\
&\cup \big\{v.A_{V_N} = a \mid v \in V_N \wedge A_{V_N}(v) = a\big\} \\
&\cup \big\{e.A_{E_N} = a \mid e \in E_N \wedge A_{E_N}(e) = a\big\}. \tag{12}
\end{aligned}$$

We denote $\mathcal{I}_N \colon (N, A_N) \mapsto (\tilde{E}_{\tilde{O}}, \tilde{A}_{\tilde{O}})$. One can easily see that $\mathcal{I}_N^{-1} \circ \mathcal{I}_N$ is an identity (transformation preserves all the information).

On the other hand, given a dataset (E_O, A_O), represented with an ontology on semantic level, inferred (undirected) network $(\tilde{N}, \tilde{A}_{\tilde{N}})$ on data level is defined with

$$\tilde{V}_{\tilde{N}} = E_O \cap E^I, \tag{13}$$
$$\tilde{E}_{\tilde{N}} = \big\{E_O^a \cap E^I \mid a \in A_O \wedge E_O^a \subseteq E_O\big\}, \tag{14}$$

and

$$\tilde{A}_{\tilde{V}_{\tilde{N}}} \colon \tilde{V}_{\tilde{N}} \rightarrow E^C \times E^A, \tag{15}$$
$$\tilde{A}_{\tilde{E}_{\tilde{N}}} \colon \tilde{E}_{\tilde{N}} \rightarrow E^R. \tag{16}$$

Instances of ontology are represented with the vertices of the network, and axioms with its edges. Classes and related data are, together with the attributes, expressed through vertex, edge attribute functions.

We denote $\mathcal{I}_O \colon (E_O, A_O) \mapsto (\tilde{N}, \tilde{A}_{\tilde{N}})$. Transformation $\mathcal{I}_O$ discards purely semantic information (e.g., related data between classes), as it cannot be represented on the data level. Thus $\mathcal{I}_O$ cannot be inverted as $\mathcal{I}_N$. However, all the data, and data related information, is preserved (e.g., individuals, classes and related data among individuals).

Due to limitations of networks, only axioms, relating at most two individuals in E_O, can be represented with the set of edges $\tilde{E}_{\tilde{N}}$ (14). When this is not sufficient, *hyper-networks* (or *hypergraphs*[1]) should be employed instead. Nevertheless, networks should suffice in most cases.

One more issue has to be stressed. Although $\mathcal{I}_N$ and $\mathcal{I}_O$ give a "common" representation of every dataset, the transformations are completely different. For instance, presume (N, A_N) and (E_O, A_O) are (given) representations of the same dataset. Then $\mathcal{I}_N(N, A_N) \neq (E_O, A_O)$ and $\mathcal{I}_O(E_O, A_O) \neq (N, A_N)$ in general – inferred ontology, network does not equal given ontology, network respectively. The former non-equation resides in the fact that network (N, A_N) contains no knowledge of the (pure) semantics within ontology (E_O, A_O); and the latter resides in the fact that $\mathcal{I}_O$ has no information of the exact representation used for (N, A_N). Still, transformations $\mathcal{I}_N$ and $\mathcal{I}_O$ can be used to manage data on a common basis.

Last, we discuss three key issues regarding an adequate data architecture, presented in Section 3. Firstly, due to variety of different data formats, a mutual representation must be employed. As the data on both data and semantic level is represented in the form of knowledge chunks (Section 3.2), every piece of data is stored in exactly the same way. This allows for common algorithms of matching and merging and makes the data easily manageable.

Furthermore, introduction of knowledge chunks naturally deals also with missing data. As each chunk is actually a set of attribute-value pairs, missing data only results in smaller chunks. Alternatively, missing data could be randomly inputted from the rest and treated as extremely uncertain or mistrustful (Section 4).

Secondly, semantical component of data should be addressed properly. Proposed architecture allows simple (related) data and also semantically enriched data. Hence no information is discarded. Moreover, appropriate transformations make all data accessible on both data and semantic level, providing for specific needs of each algorithm.

Thirdly, architecture should deal with (partially) missing and uncertain or mistrustful data, which is thoroughly discussed in the following section.

4. Trust and Trust Management

When merging data from different sources, these are often of different origin and thus their trustworthiness (or accuracy) can be questionable. For instance, personal data of participants in a traffic accident is usually more accurate in the police record of the accident, then inside participants' social network profiles. Nevertheless, an attribute from less trusted data source can still be more accurate than an attribute from more trusted one – a related status (e.g., single or married) in the record may be outdated, while such type of information is inside the social network profiles quite often up-to-date.

A complete solution for matching and merging execution should address such problems as well. A common approach for dealing with data sources that provide untrust-

---

[1]Hypergraphs are graphs, where edges can connect multiple vertices.

worthy or conflicting statements, is the use of *trust management (systems)*. These are, alongside the concept of trust, both further discussed in Sections 4.1 and 4.2.

4.1. *Definition of Trust*

Trust is a complex psychological-sociological phenomenon. Despite of, people use term trust in everyday life widely, and with very different meanings. Most common definition states that trust is an *assured reliance on the character, ability, strength, or truth of someone or something.*

In the context of computer networks, trust is modeled as a related data between entities. Formally, we define a *trust related data* as

$$\omega_E \colon E \times E \to \Sigma^E, \tag{17}$$

where E is a set of entities and Σ^E a set of all possible, numerical or descriptive, trust values. ω_E thus represents one entity's attitude towards another and is used to model trust(worthiness) of all entities in E. To this end, different trust modeling methodologies and systems can be employed, from qualitative to quantitative, e.g., Nagy (2008), Richardson (2003), Treck (2009).

We introduce trust on three different levels. First, we define trust on the level of data source, in order to represent trustworthiness of the source in general. Let S be the set of all data sources. Their trust is defined as $T_S \colon S \to [0, 1]$, where higher values of T_S represent more trustworthy source.

Second, we define trust on the level of attributes (or semantically related data) within the knowledge chunks. The trust in attributes is naturally dependent on the data source of origin, and is defined as $T_{A_s} \colon A_s \to [0, 1]$, where A_s is the set of attributes for data source $s \in S$. As before, higher values of T_{A_s} represent more trustworthy attribute.

Last, we define trust on the level of knowledge chunks. Despite the trustworthiness of data source and attributes within some knowledge chunk, its data can be (semantically) corrupted, missing or otherwise unreliable. This information is captured using trustworthiness of knowledge chunks, and again defined as $T_K \colon K \to [0, 1]$, where K is a set of all knowledge chunks. Although the trust related data (17), needed for the evaluation of trustworthiness of data sources and attributes, are (mainly) defined by the user, computation of trust in knowledge chunks can be fully automated using proper evaluation function (Section 4.2).

Three levels of trust provide high flexibility during matching and merging. For instance, attributes from more trusted data sources are generally favored over those from less trusted ones. However, by properly assigning trust in attributes, certain attributes from else less trusted data sources can prevail. Moreover, trust in knowledge chunks can also assist in revealing corrupted, and thus questionable, chunks that should be excluded from further execution.

Finally, we define trust in some particular value within a knowledge chunk, denoted *trust value T*. This is the value in fact used during merging and matching execution and is computed from corresponding trusts on all three levels. In general, T can be an

arbitrary function of T_S, T_{A_s} and T_K. Assuming independence, we calculate trust value by concatenating corresponding trusts,

$$T = T_S \circ T_{A_s} \circ T_K. \tag{18}$$

Concatenation function $\circ$ could be a simple multiplication or some fuzzy logic operation (trusts should in this case be defined as fuzzy sets).

4.2. *Trust Management*

During merging and matching execution, trust values are computed using trust management algorithm based on Richardson *et al.* (2003). We begin by assigning trust values T_S, T_{A_s} for each data source, attribute respectively (we actually assign trust related data). Commonly, only a subset of values must necessarily be assigned, as others can be inferred or estimated from the first. Next, trust values for each knowledge chunk are not defined by the user, but are calculated using the *chunk evaluation function* f_{eval} (i.e., $T_K = f_{\text{eval}}$).

An example of such function is a *density of inconsistencies* within some knowledge chunk. For instance, when attributes *Birth* and *Age* of some particular knowledge chunk mismatch, this can be seen as an inconsistency. However, one must also consider the trust of the corresponding attributes (and data sources), as only inconsistencies among trustworthy attributes should be considered. Formally, density of inconsistencies is defined as

$$f_{\text{eval}}(k) = \frac{\hat{N}_{\text{inc}}(k) - N_{\text{inc}}(k)}{\hat{N}_{\text{inc}}(k)}, \tag{19}$$

where k is a knowledge chunk, $k \in K$, $N_{\text{inc}}(k)$ the number of inconsistencies within k and $\hat{N}_{\text{inc}}(k)$ the number of all possible inconsistencies.

Finally, after all individual trusts T_S, T_{A_s} and T_K have been assigned, trust values T are computed using (18). When merging takes place and two or more data sources (or knowledge chunks) provide conflicting attribute values, corresponding to the same (resolved) entity, trust values T are used to determine actual attribute value in the resulting data source (or knowledge chunk). For further discussion on trust management during matching and merging see Section 5.

5. Matching and Merging Data Sources

Merging data from heterogeneous sources can be seen as a two-step process. The first step resolves the real world entities of abstract level, described by the data on lower levels, and constructs a mapping between the levels. This mapping is used in the second step that actually merges the datasets at hand. We denote these subsequent steps as *entity resolution* (i.e., matching) and *redundancy elimination* (i.e., merging).

Matching and merging is employed in various scenarios. As the specific needs of each scenario vary, different dimensions of variability characterize every matching and

merging execution. These dimensions are managed through the use of *contexts* (Castano *et al.*, 2010; Lapouchnian and Mylopoulos, 2009). Contexts allow a formal definition of specific needs arising in diverse scenarios and a joint control over various dimensions of matching and merging execution.

The following section discusses the notion of contexts more thoroughly and introduces different types of contexts used. Next, Sections 5.2 and 5.3 describe employed *entity resolution* and *redundancy elimination* algorithms respectively. The general framework for matching and merging is presented and formalized in Section 5.4, and discussed in Section 7.

5.1. *Contexts*

Every matching and merging execution is characterized by different dimensions of variability of the data, and mappings between. *Contexts* are a formal representation of all possible operations in these dimensions, providing for specific needs of each scenario. Every execution is thus characterized with the contexts it defines (Fig. 4), and can be managed and controlled through their use.

The idea of contexts originates in the field of requirements engineering, where it has been applied to model domain variability (Lapouchnian and Mylopoulos, 2009). It has just recently been proposed to model also variability of the matching execution (Castano *et al.*, 2010). Our work goes one step further as it introduces contexts, not bounded only to user or scenario specific dimensions, but also data related and trust contexts.

Formally, we define a context C as

$$C: D \rightarrow \{\text{true}, \text{false}\}, \tag{20}$$

where D can be any simple or composite domain. A context simply limits all possible values, attributes, related data, knowledge chunks, datasets, sources or other, that are
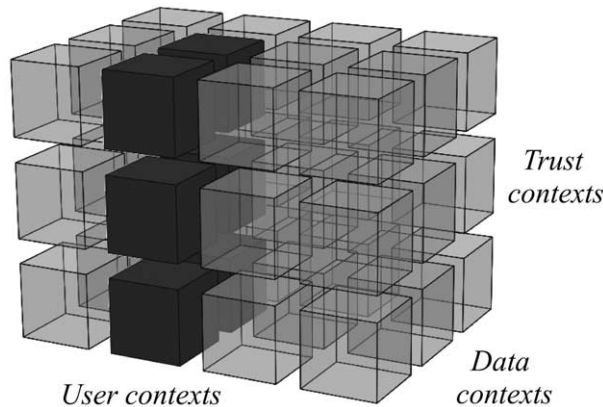

Fig. 4. Characterization of merging and matching execution defining one context in user dimension, two contexts in data dimension and all contexts in trust dimension.

considered in different parts of matching and merging execution. Despite its simple definition, a context can be a complex function. It is defined on any of the architecture levels, preferably on all. Let C_A, C_S and C_D represent the same context on abstract, semantic and data level respectively. The joint context is defined as

$$C_J = C_A \wedge C_S \wedge C_D. \tag{21}$$

In the case of missing data (or contexts), only appropriate contexts are considered. Alternatively, contexts could be defined as fuzzy sets, to address also the noisiness of data. In that case, a fuzzy AND operation should be used to derive joint context C_J.

We distinguish between three types of contexts due to different dimensions characterized (Fig. 4).

user User or scenario specific contexts are used mainly to limit the data and control the execution. This type coincides with dimensions identified in Castano *et al.* (2010). An example of user context is a simple selection or projection of the data.

data Data related contexts arise from dealing with related or semantic data, and various formats of data. Missing or corrupted data can also be managed through the use of these contexts.

trust Trust and data uncertainty contexts provide for an adequate trust management and efficient security assurance between and during different phases of execution. An example of trust context is a definition of required level of trustworthiness of data or sources.

Detailed description of each context is out of scope of this paper. For more details on (user) contexts see Castano *et al.* (2010).

5.2. *Entity Resolution*

First step of matching and merging execution is to resolve the real world entities on abstract level, described by the data on lower levels. Thus a mapping between the levels (entities) is constructed and used in consequent merging execution. Recent literature proposes several state-of-the-art approaches for entity resolution (e.g., Ananthakrisha *et al.*, 2002; Bhattacharya and Getoor, 2004, 2007; Dong *et al.*, 2005; Kalashnikov and Mehrotra, 2006). A naive approach is a simple pairwise comparison of attribute values among different entities. Although, such an approach could already be sufficient for flat data, this is not the case for network data, as the approach completely discards related data between the entities. For instance, when two entities are related to similar entities, they are more likely to represent the same entity. However, only the attributes of the related entities are compared, thus the approach still discards the information if related entities resolve to the same entities – entities are even more likely to represent the same entities when their related entities resolve to, not only similar, but the same entities. An approach that uses this information, and thus resolves entities altogether (in a collective fashion), is denoted *collective* (related) entity resolution algorithm.

We employ a state-of-the-art *(collective) related data clustering* algorithm proposed in Bhattacharya and Getoor (2007). To further enhance the performance, algorithm is semantically elevated and adapted to allow for proper and efficient trust management.

The algorithm is actually a greedy agglomerative clustering (Algoritm 1). Entities (on lower levels) are represented as a group of clusters C, where each cluster represents a set of entities that resolve to the same entity on abstract level. At the beginning, each (lower level) entity resides in a separate cluster. Then, at each step, the algorithm merges two clusters in C that are most likely to represent the same entity (most *similar* clusters). When the algorithm unfolds, C holds a mapping between the entities on each level (i.e., maps entities on lower levels through the entities on abstract level).

During the algorithm, similarity of clusters is computed using a *joint similarity measure* (28), combining *attribute*, *related data* and *semantic similarity*. First is a basic pairwise comparison of attribute values, second introduces related information into the computation of similarity (in a collective fashion), while third represents semantic elevation of the algorithm.

Let $c_i, c_j \in C$ be two clusters of entities. Using knowledge chunk representation, attribute cluster similarity is defined as

$$\text{sim}_A(c_i, c_j) = \sum_{k_{i,j} \in c_{i,j} \wedge a \in k_{i,j}} \text{trust}(k_i.a, k_j.a) \text{sim}_A(k_i.a, k_j.a), \tag{22}$$

where $k_{i,j} \in K$ are knowledge chunks, $a \in A_s$ is an attribute and $\text{sim}_A(k_i.a, k_j.a)$ similarity between two attribute values. (Attribute) similarity between two clusters is thus defined as a weighted sum of similarities between each pair of values in each knowledge chunk. Weights are assigned due to trustworthiness of values – trust in values $k_i.a$ and $k_j.a$ is computed using

$$\text{trust}(k_i.a, k_j.a) = \min \big\{ T(k_i.a), T(k_j.a) \big\}. \tag{23}$$

Hence, when even one of the values is uncertain or mistrustful, similarity is penalized appropriately, to prevent matching based on (likely) incorrect information.

For computation of similarity between actual attribute values $\text{sim}_A(k_i.a, k_j.a)$ (22), different measures have been proposed. *Levenshtein* (1996) *distance* measures edit distance between two strings – number of insertions, deletions and replacements that traverse one string into the other. Another class of similarity measures are *TF-IDF*[2]-based measures, e.g., *Cos TF-IDF* and *Soft TF-IDF* (Cohen *et al.*, 2003; Moreau *et al.*, 2008). They treat attribute values as a bag of words, thus the order of words in the attribute has no impact on the similarity. Other attribute measures are also *Jaro* (Jaro, 1989) and *Jaro–Winkler* (Winkler, 1990) that count number of matching characters between the attributes.

Different similarity measures prefer different types of attributes. *TF-IDF*-based measures work best with longer strings (e.g., descriptions), when other prefer shorter strings (e.g., names). For numerical attributes, an alternative measure has to be employed (e.g., simple evaluation, followed by a numerical comparison). Therefore, when computing attribute similarity for a pair of clusters, different attribute measures are used with different attributes (22).

---

[2]*Term Frequency-Inverse Document Frequency.*

Using data level representation, we define a *neighborhood* for vertex $v \in V_N$ as

$$\mathrm{nbr}(v) = \big\{v_n|\, v_n \in V_N \wedge \{v, v_n\} \in E_N\big\}, \tag{24}$$

and cluster $c \in C$ as

$$\mathrm{nbr}(c) = \big\{c_n|\, c_n \in C \wedge v \in c \wedge c_n \cap \mathrm{nbr}(v) \neq \emptyset\big\}. \tag{25}$$

Neighborhood of a vertex is defined as a set of connected vertices. Similarly, neighborhood of a cluster is defined as a set of clusters, connected through the vertices within.

For a (collective) related similarity measure, we adapt a *Jaccard coefficient* (Bhattacharya and Getoor, 2007) measure for trust-aware (related) data. *Jaccard coefficient* is based on *Jaccard index* and measures the number of common neighbors of two clusters, considering also the size of the clusters' neighborhoods – when the size of neighborhoods is large, the probability of common neighbors increases. We define

$$\mathrm{sim}_R(c_i, c_j) = \frac{\sum_{c_n \in \mathrm{nbr}(c_i) \cap \mathrm{nbr}(c_j)} \mathrm{trust}(e_{in}^T, e_{jn}^T)}{|\mathrm{nbr}(c_i) \cup \mathrm{nbr}(c_j)|} \tag{26}$$

where e_{in}^T, e_{jn}^T is the most trustworthy edge connecting vertices in c_n and c_i, c_j respectively (for the computation of $\mathrm{trust}(e_{in}^T, e_{jn}^T)$, a knowledge chunk representation of e_{in}^T, e_{jn}^T is used). (Related data) similarity between two clusters is defined as the size of a common neighborhood (considering also the trustworthiness of connecting related data), decreased due to the size of clusters' neighborhoods. Entities related to a relatively large set of entities that resolve to the same entities on abstract level, are thus considered to be similar.

Alternatively, one could use some other similarity measure like *Adar–Adamic* (2001) *similarity*, random walk measures, or measures considering also the ambiguity of attributes or higher order neighborhoods (Bhattacharya and Getoor, 2007).

For the computation of the last, semantic, similarity, we propose a random walk like approach. Using a semantic level representation of clusters $c_i, c_j \in C$, we do a number of random assumptions (queries) over underlying ontologies. Let N_{ass} be the number of times the consequences (results) of the assumptions made matched, $\tilde{N}_{\mathrm{ass}}$ number of times the consequences were undefined (for at least one ontology) and $\hat{N}_{\mathrm{ass}}$ the number of all assumptions made. Furthermore, let N_{ass}^T be the trustworthiness of ontology elements used for reasoning in assumptions that matched (computed as a sum of products of trusts on the paths of reasoning, similar as in (23). Semantic similarity is then defined as

$$\mathrm{sim}_S(c_i, c_j) = \frac{N_{\mathrm{ass}}^T(c_i, c_j)}{\hat{N}_{\mathrm{ass}}(c_i, c_j) - \tilde{N}_{\mathrm{ass}}(c_i, c_j)}. \tag{27}$$

Similarity represents the trust in the number of times ontologies produced the same consequences, not considering assumptions that were undefined for some ontology. As the expressiveness of different ontologies vary, and some of them are even inferred from

network data, many of the assumptions could be undefined for some ontology. Still, for $\hat{N}_{\mathrm{ass}}(c_i, c_j) - \tilde{N}_{\mathrm{ass}}(c_i, c_j)$ large enough, (27) gives a good approximation of semantic similarity.

Using attribute, related and semantic similarity (22), (26) and (27) we define a joint similarity for two clusters as

$$
\begin{aligned}
\mathrm{sim}(c_i, c_j) = {} & \frac{1}{\delta_A + \delta_R + \delta_S} \\
& \times \big(\delta_A \, \mathrm{sim}_A(c_i, c_j) + \delta_R \, \mathrm{sim}_R(c_i, c_j) + \delta_S \, \mathrm{sim}_S(c_i, c_j) \big),
\end{aligned} \tag{28}
$$

where δ_A, δ_R and δ_S are weights, set due to the scale of related and semantical information within the data. For instance, setting $\delta_R = \delta_S = 0$ reduces the algorithm to a naive pairwise comparison of attribute values, which should be used when no related or semantic information is present.

Finally, we present the collective entity resolution Algoritm 1. First, the algorithm initializes clusters C and priority queue of similarities Q, considering the current set of clusters (lines 1–5). Each cluster represents at most one entity as it is composed out of a single knowledge chunk. Algorithm then, at each iteration, retrieves currently the most

---

Algorithm 1. Collective entity resolution

```
 1: Initialize clusters as C = {{k}| k ∈ K}
 2: Initialize priority queue as Q = ∅
 3: for c_i, c_j ∈ C and sim(c_i, c_j) ⩾ θ_S do
 4:     Q.insert(sim (c_i, c_j), c_i, c_j)
 5: end for
 6: while Q ≠ ∅ do
 7:     (sim(c_i, c_j), c_i, c_j) ← Q.pop() {Most similar.}
 8:     if sim(c_i, c_j) < θ_S then
 9:         return C
10:     end if
11:     C ← C − {c_i, c_j} ∪ {c_i ∪ c_j} {Matching.}
12:     for (sim(c_x, c_k), c_x, c_k) ∈ Q and x ∈ {i, j} do
13:         Q.remove(sim(c_x, c_k), c_x, c_k)
14:     end for 15:     for c_k ∈ C and sim(c_i ∪ c_j, c_k) ⩾ θ_S do
16:         Q.insert(sim(c_i ∪ c_j, c_k), c_i ∪ c_j, c_k)
17:     end for
18:     for c_n ∈ nbr(c_i ∪ c_j) do
19:         for c_k ∈ C and sim(c_n, c_k) ⩾ θ_S do
20:             Q.insert(sim(c_n, c_k), c_n, c_k) {Or update.}
21:         end for
22:     end for
23: end while
24: return C
```

similar clusters and merges them (i.e., matching of resolved entities), when their similarity is greater than threshold θ_S (lines 7–11). As clusters are stored in the form of knowledge chunks, matching in line 11 results in a simple concatenation of chunks. Next, lines 12–17 update similarities in the priority queue Q, and lines 18–22 insert (or update) also neighbors' similarities (required due to related similarity measure). When the algorithm terminates, clusters C represent chunks of data resolved to the same entity on abstract level. This mapping between the entities (i.e., their knowledge chunk re presentations) is used to merge the data in the next step.

Threshold θ_S represents minimum similarity for two clusters that are considered to represent the same entities. Optimal value should be estimated from the data.

Three more aspects of the algorithm ought to be discussed. Firstly, pairwise comparison of all clusters during the execution of the algorithm is computationally expensive, specially in early stages of the algorithm. Bhattacharya and Getoor (2007) propose an approach in which they initially find groups of chunks that could possibly resolve to the same entity. In this way, the number of comparisons can be significantly decreased.

Secondly, due to the nature of (collective) related similarity measures, they are ineffective when none of the entities has already been resolved (e.g., in early stages of the algorithm). As the measure in (26) counts the number of common neighbors, this always evaluates to 0 in early stages (in general). Thus relative similarity measures should be used after the algorithm has already resolved some of the entities, using only attribute and semantic similarities.

Thirdly, in the algorithm we implicitly assumed that all attributes, (semantic) related data and other, have the same names or identifiers in every dataset (or knowledge chunk). Although, we can probably assume that all attributes within datasets, produced by the same source, have same and unique names, this cannot be generalized.

We propose a simple, yet effective, solution. The problem at hand could be denoted *attribute resolution*, as we merely wish to map attributes between the datasets. Thus we can use the approach proposed for entity resolution. Entities are in this case attributes that are compared due to their names, and also due to different values they hold; and related data between entities (attributes) represent co-occurrence in the knowledge chunks. As certain attributes commonly occur with some other attributes, this would further improve the resolution.

Another possible improvement is to address also the attribute values in a similar manner. As different values can represent the same underlying value, *value resolution*, done prior to attribute resolution, can even further improve the performance.

5.3. *Redundancy Elimination*

After the entities, residing in the data, have been resolved (Section 5.2), the next step is to eliminate the redundancy and merge the datasets at hand. This process is somewhat straightforward as all data is represented in the form of knowledge chunks. Thus we merely need to merge the knowledge chunks, resolved to the same entity on abstract level. Redundancy elimination is done entirely on semantic level, to preserve all the knowledge inside the data.
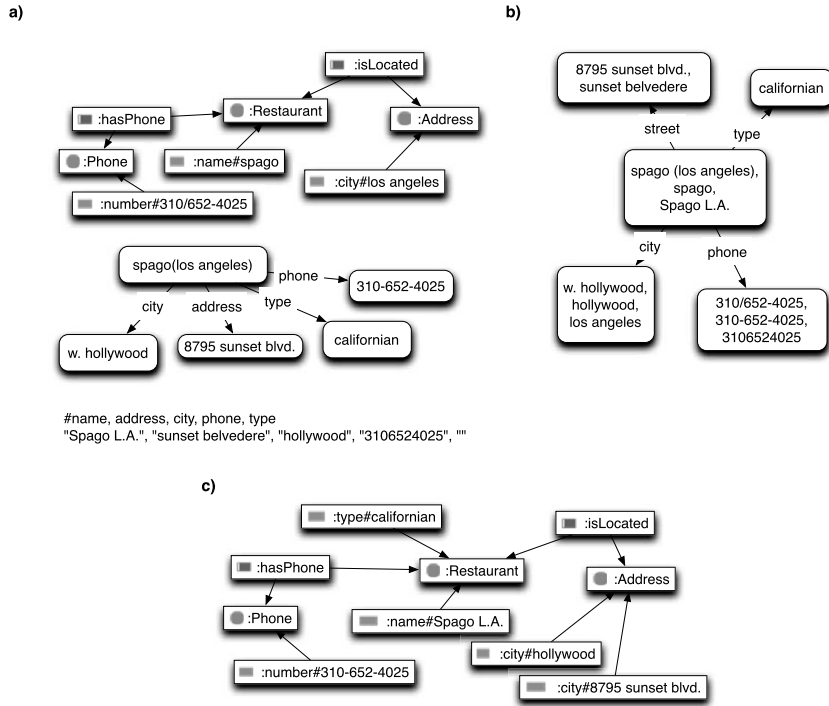
Fig. 5. Entity resolution and redundancy elimination on three knowledge chunks (Section 3.2). (a) Input data in a form of ontology (Fig. 2), network and attribute values. (b) Cluster network obtained with entity resolution (i.e., matching). (c) Final ontology obtained after redundancy elimination and appropriate postprocessing.

When knowledge chunks hold disjoint data (i.e., attributes), they can simply be concatenated together. However, commonly various chunks would provide values for the same attribute and, when these values are inconsistent, they need to be handled appropriately. A naive approach would count only the number of occurrences of some value, when we consider also their trustworthiness, to determine the most probable value for each attribute.

Let $c \in C$ be a cluster representing some entity on abstract level (resolved in the previous step), let $k_1, k_2 \ldots k_n \in c$ be its knowledge chunks and let k^c be the *merged knowledge chunk*, we wish to obtain. Furthermore, for some attribute $a \in A$., let X^a be a random variable measuring the true value of a and let X_i^a be the random variables for a in each knowledge chunk it occurs (i.e., $k_i.a$). Value of attribute a for the merged knowledge chunk k^c is then defined as

$$\arg \max_v P\Big(X^a = v | \bigwedge_i X_i^a = k_i.a\Big). \tag{29}$$

Each attribute is thus assigned the most probable value, given the evidence observed (i.e., values $k_i.a$). By assuming pair-wise independence among X_i^a (conditional on X^a) and

---

Algorithm 2. Redundancy elimination

1: Initialize knowledge chunks K^C

2: **for** $c \in C$ **and** $a \in A$. **do**

3:     $k^c.a = \arg \max_v \prod_{k \in c \wedge k.a = v} T(k.a) \prod_{k \in c \wedge k.a \neq v} 1 - T(k.a)$

4: **end for**

5: **return** K^C

---

uniform distribution of X^a (29) simplifies to

$$\arg \max_v \prod_i P\big(X_i^a = k_i.a | X^a = v\big). \tag{30}$$

Finally, conditional probabilities in (30) are approximated with trustworthiness of values,

$$P(X_i^a | X^a) \approx \begin{cases} T(k_i.a) & \text{for } k_i.a = v, \\ 1 - T(k_i.a) & \text{for } k_i.a \neq v, \end{cases} \tag{31}$$

hence

$$k^c.a = \arg \max_v \prod_{k_i.a = v} T(k_i.a) \prod_{k_i.a \neq v} 1 - T(k_i.a). \tag{32}$$

Only knowledge chunks (Section 3.2) containing attribute a are considered.

In the following we present the proposed redundancy elimination Algoritm 2.

The algorithm uses knowledge chunk representation of semantic level. First, it initializes merged knowledge chunks $k^c \in K^C$. Then, for each attribute $k^c.a$, it finds the most probable value among all given knowledge chunks (line 3). When the algorithm unfolds, knowledge chunks K^C represent a merged dataset, with resolved entities and eliminated redundancy. Each knowledge chunk k^c corresponds to unique entity on abstract level, and each attribute holds the most trustworthy value.

At the end, only the data that was actually provided by some data source, should be preserved. Thus all inferred data (through $\mathcal{I}_N$ or $\mathcal{I}_O$; Section 3.3) is discarded, as it is merely an artificial representation needed for (common) entity resolution and redundancy elimination. Still, all provided data and semantical information is preserved and properly merged with the rest. Hence, although redundancy elimination is done on semantic level, resulting dataset is given on both data and semantic level (that complement each other).

Last, we discuss the assumptions of independence among X_i^a and uniform distribution of X^a. Clearly, both assumptions are violated, still the former must be made in order for the computation of most probable value to be feasible. However, the latter can be eliminated when distribution of X^a can be approximated from some large-enough dataset.

5.4. *General Framework*

Proposed entity resolution and redundancy elimination algorithms (Section 5.2, 5.3) are integrated into a general *framework* for matching and merging (Fig. 6). Framework rep-
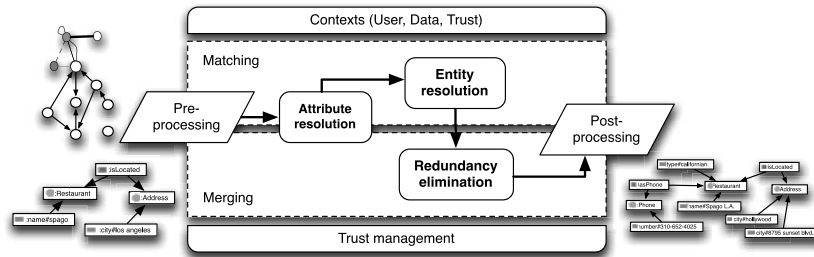
Fig. 6. General framework for matching and merging data from heterogeneous sources.

resents a complete solution, allowing a joint control over various dimensions of matching and merging execution. Each component of the framework is briefly presented in the following, and further discussed in Section 7.

Initially, data from various sources is preprocessed appropriately. Every network or ontology is transformed into a knowledge chunk representation and, when needed, also inferred on an absent architecture level (Section 3.3). After preprocessing is done, all data is represented in the same, easily manageable, form, allowing for common, semantically elevated, subsequent analyses.

Prior to entity resolution, attribute resolution is done (Section 5.2). The process resolves and matches attributes in the heterogeneous datasets, using the same algorithm as for entity resolution. As all data is represented in the form of knowledge chunks, this actually unifies all the underlying networks and ontologies.

Next, proposed entity resolution and redundancy elimination algorithms are employed (Section 5.2, 5.3). The process thus first resolves entities in the data, and then uses this information to eliminate the redundancy and to merge the datasets at hand. Algorithms explore not only the related data, but also the semantics behind it, to further improve the performance.

Last, postprocessing is done, in order to discard all artificially inferred data and to translate knowledge chunks back to the original network or ontology representation (Section 3). Throughout the entire execution, components are jointly controlled through (defined) user, data and trust contexts (Section 5.1). Furthermore, contexts also manage the results of the algorithms, to account for specific needs of each scenario.

Every component of the framework is further enhanced, to allow for proper trust management, and thus also for efficient security assurance. In particular, all the similarity measures for entity resolution are trust-aware, moreover, trust is even used as a primary evidence in the redundancy elimination algorithm. The introduction of trust-aware and security-aware algorithms represents the main novelty of the proposition.

6. Experiments

In the following sections we demonstrate the framework's (Fig. 6) most important parts on several real-world datasets, designed for entity resolution tasks and discuss the results.

The part of attribute resolution and redundancy elimination evaluation is shown like a case study because to our knowledge, no tagged data combining all results we need, exists.

The demonstration is done with respect to semantic elevation, semantic similarity and trust management contexts (Section 5.1). We do not fit methods for the datasets to achieve superior performance, but rather focus on the increase of accuracy when using each of the contexts. In the following we present the datasets, explain used metrics, show the results and discuss them. The used datasets and full source code is publicly available[3].

6.1. *Datasets*

We consider five testbeds of four different domains to simulate real-life matching tasks. Each data source introduces many data quality problems, in particular duplicate references, heterogeneous representations, misspellings or extraction errors.

The CiteSeer dataset used is a cleaned version from Bhattachrya and Getoor (2007)[4], others were presented and evaluated against entity resolution algorithms by Köpcke *et al.* (2010)[5].

CiteSeer dataset contains 1504 machine learning documents with 2892 author references to 1165 author entities. The only attribute information available is name for authors and title for documents.

DBLP-ACM dataset consists of two well-structured bibliographic data sources from DBLP and ACM with 2616 and 2294 references to 2224 document entities. Each reference contains values for title, authors, venue and publication year of respective scientific paper.

Restaurants dataset contains 864 references to 754 restaurant entities. Most of the references contain values for name, address, city, phone number and type of certain restaurant.

AbtBuy is an e-commerce dataset with extracted data from `Abt.com` and `Buy.com`. They contain 1081 and 1092 references to 1097 different products. Each product reference is mostly represented by product name, manufacturer and often missing description and price values.

Affiliations dataset consists of 2260 references to 331 organizations. The only attribute value per record is an organization name, which can be written in many possible ways (i.e., full, part name or abbreviation).

6.2. *Attribute Resolution*

As a part of semantic elevation, the input datasets must be aligned by *attribute-value* pairs to achieve a mutual representation.

As mentioned in Section 5.2, an entity resolution algorithm could be used to merge appropriate attributes. To better solve the problem in general, we propose the following similarity functions:

---

[3]`http://zitnik.si/mediawiki/index.php?title=Software.`
[4]`http://www.cs.umd.edu/projects/linqs/projects/er/DATA/citeseer.dat.`
[5]`http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/`
`benchmark_datasets_for_entity_resolution.`

Table 1

Attribute names, used to test attribute resolution approaches on *DBLP-ACM* dataset

id	Attribute names			
1	title	venue	year	authors
2	titl	venue	year	author
3	title	venue	yr	writers
4	attr1	attr2	attr3	attr4

ExactMatch. The simplest version. Attribute names must match exactly.

SimilarityMatch. Every two attributes with score above the selected threshold, are matched (We use *Jaro–Winkler* (Winkler, 1990) metric with threshold of 0.95). This is typical pairwise entity resolution approach.

SimilarityMatch+. In addition to previous function, it considers synonyms when comparing two attribute names (Synsets from semantic lexicon *Wordnet* (Miller, 1995) are used.). Real-life datasets along with attribute names are created by people and that is why attributes over different datasets are supposed to be synonyms.

DomainMatch. Same attribute values contain similar data format. Leveraging this information we extract selected features and match the most similar attributes across datasets. (A simple example is calculating the average number of words per attribute values.)

OntologyMatch. Using ontologies, additional semantic information is included. If all input datasets are semantically described using ontologies, related data types *sameAs* or *seeAlso*, possible hierarchy of *subclasses*, included rules and axioms can be additionally used for matching. When none of this apply, previous procedures must be employed.

As our datasets mostly consist of 2 different already aligned sources, we have chosen some additional attribute names for *DBLP-ACM* dataset manually. Altered values are shown in Table 1. Due to space limitations, we just presentively discuss the results. First line are the original attribute names and next three lines are changed to show success of proposed matchers. Pair (1, 2) is successfully solved by *SimilarityMatch*. The difference between values is limited to misspellings and small writing errors. Pair (1, 3) is a bit more difficult. Values *authors* and *writers* or *year* and *yr* cannot be matched by similarity. As they are synonyms, *SimilarityMatch+* can match them. Pair (1, 4) values are completely different and it is completely useless to check name pairs. The *DomainMatch* technique correctly matches the attributes by considering attribute values format.

6.3. *Entity Resolution*

In this section we first discuss the selected entity resolution algorithm and then show the increase in correctly matched values using semantic similarity. We implemented the algorithm, proposed in Bhattacharya and Getoor (2007), which is well described in Section 5.2 and presented as Algoritm 1.

In addition to the standard blocking techniques of partially string matching we added similarity, n-gram blocking and also enabled the option of fuzzy blocking. Standard approach is used on *AbtBuy*, *CiteSeer* and *DBLP-ACM* datasets. Similarity blocking adds an instance to a block if the similarity score with the representative reference of the block is above the defined threshold. This type of blocking was used with the *Restaurants* dataset using 0.3 threshold for *name* and 0.7 for *phone* attribute. At *Affiliations* dataset we use n-gram blocking with at least 4–6-gram matches.

We use secondstring (Cohen *et al.*, 2003) library for all basic similarity measures implementations. At bootstrapping and clustering we use JaroWinkler with TFIDF and manual weights as an attribute metric. Promising general results were achieved also using n-gram and Level2JaroWinkler metric. As a related similarity we use k-Neighbours at bootstrapping and modified JaccardCoefficient at clustering. The modification just aligns the match result x using function $f(x) = -(-x+1)^{10} + 1$, because similarity pairs instead of typical sets are checked.

The most important parameters that need to be selected are similarity alpha α and merge threshold θ_S. Both values were selected subjectively and not dataset – specific. We set similarity alpha to $\alpha = 0.85$, which results in weighting attribute metric to $\delta_A = \alpha$ and related data metric to $\delta_R = 1 - \alpha$. Matching accuracy using different similarity alphas is shown on Fig. 7. As it can be seen from the figure, some datasets contain a lot of disambiguate values, which results in very low F-score at α set to 1.

Merge threshold in our solution is set to 0.95. Testing the threshold at different values after bootstrapping is presented on Fig. 8 and after clustering on Fig. 9. It is possible to see the effect of iterative matching and related data metric from the Fig. 9, which improves the final results. During testing these parameters, no semantic measure was used yet.

Due to optimization, our implementation updates or possibly inserts only neighbour pairs of matched clusters into priority queue during clustering. The accuracy when checking only neighbours remains unchanged. Therefore the cluster $c_k \in \mathrm{nbr}(c_i \cup c_j)$ at the 19th line of Algoritm 1.
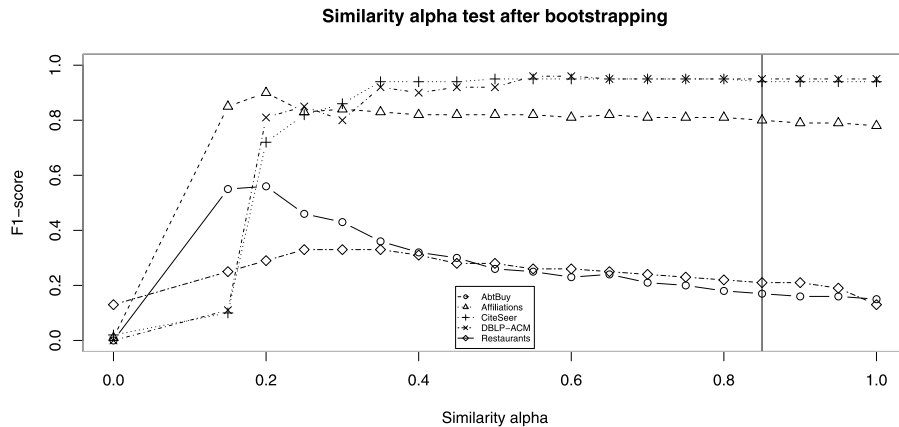
Similarity alpha test after bootstrapping

Fig. 7. Comparison of entity resolution results after bootstrapping according to similarity alpha.
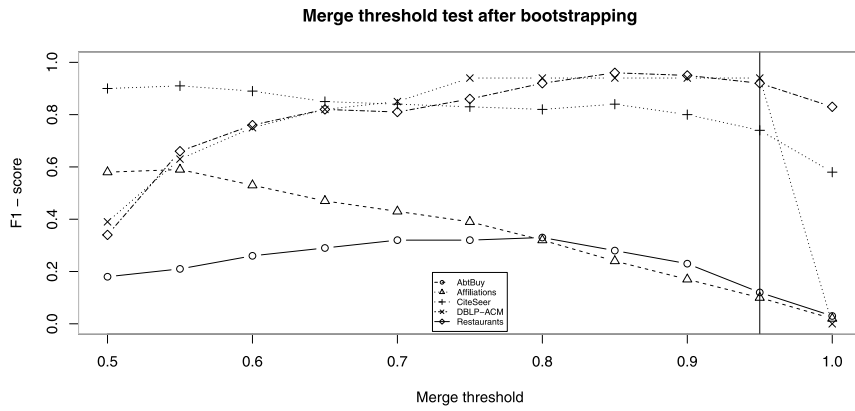
Merge threshold test after bootstrapping

Fig. 8. Comparison of entity resolution results after bootstrapping according to θ_S merge threshold.
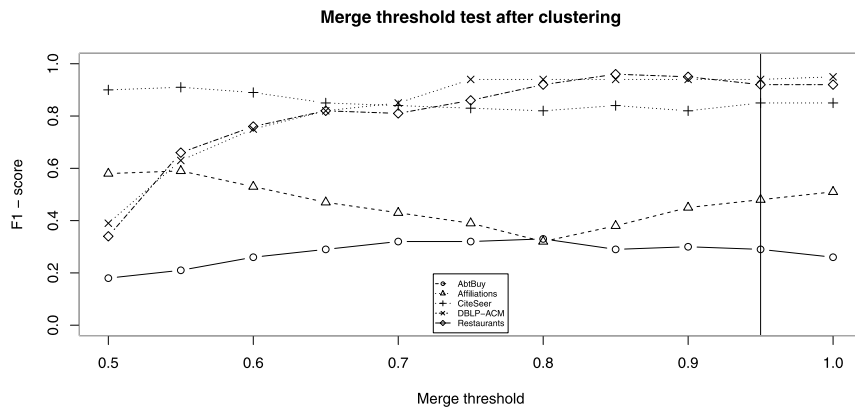
Merge threshold test after clustering

Fig. 9. Comparison of entity resolution results after clustering according to θ_S merge threshold.

On Figs. 10 and 11 we present the increase of success in matching using semantic similarity (27). We set semantic similarity weight δ_s to 4 based on some preliminary experiments. Bhattacharya and Getoor (2007) adjusted Adar (Adamic and Adar, 2001) similarity metric to better support values (e.g., author names) disambiguation. It learns an ambiguity function after checking the whole set of values in the dataset, similar to TF-IDF approach. This metric better models names, but does not use semantics, like identifying first or last name, product codes or specific parts of given value. Semantic similarity should model the human reasoning whether to match two values or not. For better understanding the meaning of semantic similarity, we present few examples, used in the experiment:

Name metric. This is our the most general similarity metric. It models typical value matching by splitting it into tokens, identifying the value with less information and comparing it to other value's tokens by *startWith* or similarity metric. It also checks and matches abbreviations. For example, every pair of names "William Cohen",

"W. Cohen", "W.W. Cohen" or "Cohen" must have maximum semantic similarity. Similar applies to "Arizona State Univ., Tempe, AZ", "Arizona State University" and "Arizona State University, Arizona" where using string similarity metric yields low values. Name metric is used for organization name matching on *Affilation* dataset, author name matching on *CiteSeer* and phone and restaurant name matching on *Restaurants* dataset.

Number metric. Number metric identifies numeric values and matches them according to difference in values. It is used on *DBLP-ACM* dataset at publication year matching.

Product metric. Product metric is designed to match products, which sometimes contain serial numbers or codes. These codes are commonly represented as a sequence of numbers and/or letters. In addition to code matching, it integrates Name metric with minimum k token match score. An example of matching two products is "Toshiba 40' Black Flat Panel LCD HDTV – 40RV525U" and "Toshiba 40RV525U – 40' Widescreen 1080p LCD HDTV w/Cinespeed – Piano Black" where it is very hard to identity pair without code detection. Product metric is used on *AbtBuy* dataset for product name matching.

Restaurant metric. This metric is specific to *Restaurants* dataset. It supposes attributes name and phone or location are scored above the threshold to match.

Title metric. Titles are sometimes shrinked, have some words replaced with synonyms or refer to papers, written in more parts. This metric improves matching titles on *DBLP-ACM dataset*.
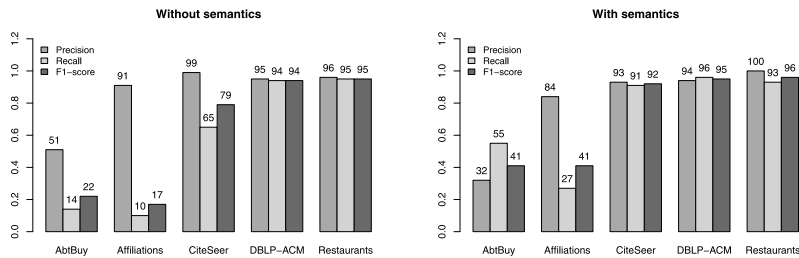

Fig. 10. Comparison of entity resolution results after bootstrapping without and with using semantic similarity.
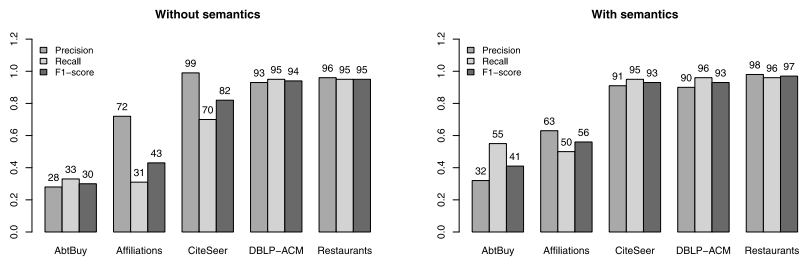

Fig. 11. Comparison of entity resolution results after clustering without and with using semantic similarity.

The results on Figs. 10 and 11 show the increase of matching accuracy by employing semantic similarity measure. The results on *AbtBuy* dataset are increased by 11% after clustering. Recall is significantly higher, but precision falls down. It is interesting that using semantics, same result is achieved immediately after bootstrapping, which shows good work at blocking. On *Affiliations*, the precision lowers, but employing semantics, more organizations with different name representations are resolved. *CiteSeer* gains more than 10% in recall and F-score and also keeps all measures above 90%. At *DBLP-ACM* dataset, the differences are not very significant, but use of semantics still shows minor improvements. After bootstrapping at *CiteSeer* dataset it is interesting semantic similarity helps achieving 100% precision and a little improves the final result.

Experimenting using only semantic similarity metric gave worse results than including also attribute one. This is because our semantic similarities focus on semantics and not on misspelled or disambiguated data on lexical level. *Restaurant* dataset for example contains examples unsolvable even for a man without background knowledge. In the case of *AbtBuy* dataset even more knowledge would not work as name and product description is too general to match on some examples. Number match metric could be applied also on it, but one of the datasets barely contains a product's price.

6.4. *Redundancy Elimination*

The last step before postprocessing is merging knowledge chunks matched in clusters at entity resolution.

Merging is done entirely using trust management. In Section 4 we define trust on levels of data source, knowledge chunk and value. As trust cannot be easily initialized, we select the appropriate cluster representative using trust of value only. Therefore we implemented the calculation of trust value for Algoritm 2 in the following ways:

Random. Random value is selected as the representative.

Naive. Value that occurs the most time is selected as the representative.

Naive+. The representative is selected as the maximum similar value to all others. Let c be a cluster of matched values, k value in cluster and Sim appropriate similarity function. Then the value is selected according to (33). As similarity function we use *Jaro–Winkler*.

$$\text{Representative}(c) = \arg\max_v \sum_{k \in c \wedge k \neq v} \text{Sim}(k, v). \tag{33}$$

Trust. Intuitively, a value is trustworthy if it yields many search results on the internet. This is not exactly true as for example the number of search results for "A. N." is much higher comparing to "Andrew Ng". By investigating some person name – based test searches, we expect the number of search hits decreases a lot if the word is misspelled. We denote $N_{\text{hits}}(v)$ as the number of hits for value v. Let $N_{\text{nhits}}(v)$ be number of hits for a value of v with some noise added. We set m to 5 and change 4 letters or numbers randomly. The trust is calculated as in (34) and as the result,

the maximum trust value is selected.

$$\text{Trust}(v) = 1 - \frac{\sum_{1 \leqslant i \leqslant m} N_{\text{nhits}}(v)}{m \cdot N_{\text{hits}}(v)}. \tag{34}$$

During experiments, random clusters, having more than 10 values of specific attribute were selected for redundancy elimination. Using clusters with multiple values, the results are more representative because it is harder to select the right value. In each cluster, we add noise to a portion of values. So, one of the non-noise values is expected to be returned as a result of redundancy elimination because they certainly better represent the entity and this is taken as a measure of classification accuracy.

Author name attribute redundancy elimination results are presented on Fig. 12 and in Table 2. The trust measure achieves better results comparing to others. It is expected
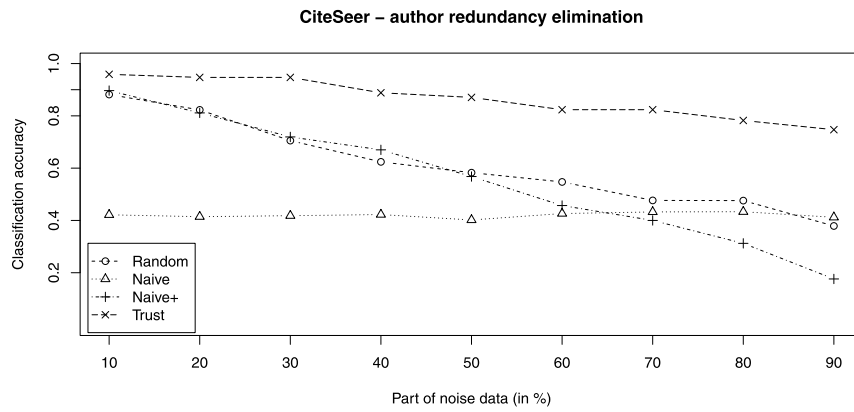

Fig. 12. Redundancy elimination on CiteSeer dataset for 34 random clusters (as a result of ER), containing author names.

Table 2

Redundancy elimination classification accuracy on 34 random author name clusters from CiteSeer dataset. The trust algorithm was repeated 10 times and others 100 times.

Noise	Random		Naive		Naive+		Trust	
	Mean	Std. d.	Mean	Std. d.	Mean	Std. d.	Mean	Std. d.
90%	0.37	0.078	0.41	0.083	0.17	0.063	**0.74**	0.084
80%	0.47	0.074	0.43	0.067	0.31	0.074	**0.78**	0.067
70%	0.47	0.069	0.43	0.075	0.39	0.070	**0.82**	0.046
60%	0.54	0.064	0.42	0.082	0.45	0.076	**0.82**	0.036
50%	0.58	0.062	0.40	0.084	0.56	0.081	**0.87**	0.033
40%	0.62	0.067	0.42	0.071	0.67	0.071	**0.88**	0.038
30%	0.70	0.062	0.41	0.083	0.71	0.075	**0.94**	0.032
20%	0.82	0.038	0.41	0.078	0.81	0.061	**0.94**	0.038
10%	0.88	0.038	0.42	0.063	0.89	0.050	**0.95**	0.044

for accuracy to be inversely proportional to level of noise, but the classification accuracy of the trust is above 70% even with 90% of noise in data. As we see, the trust measure outperforms other approaches throughout the test. The naive measure gives almost constant accuracy at all times. Naive+ approach performs vey bad by increasing the number of noise values. The reason it works better than naive at low noise levels is that there are many similar or equal values in cluster, but at higher levels, the majority of values are quite different. It's results are similar to the random measure. Random approach results are expected, maybe even too good with clusters of a lot of noise. When having no knowledge of cluster values, performance of redundancy elimination would equal to random approach.

The experiment shows it may be easy to get useful redundancy eliminator for specific types of values, but the solution remains to initialize trust levels across the domain and update them continuously during system's lifetime.

6.5. *Experiments Summary*

We presented some experiments on the attribute, entity resolution and redundancy elimination components of the proposed general *framework* for matching and merging (Fig. 6).

As first, attribute resolution matches the datasets to the same semantic representation (Section 6.2). When datasets are not appropriately matched, missing attribute pairs cannot even be compared or wrong values are considered. Therefore, further matching strongly depends on attribute resolution result.

Second, we showed entity resolution improves if additional semantic similarity measure is used (Section 6.3). Semantic similarity is attribute type-specific and cannot be defined in general. Thus, a number of metrics could be predefined and then selected for each attribute type.

Third, input to redundancy elimination are clusters as a result from entity resolution (Section 6.4). For author names, we showed the search engine results as a value of trust, can help us determine the most appropriate value. Also, this component's results strongly depend on matched clusters results as only one value within specific cluster can be selected.

To summarize, best evaluation measuring interdependence between components could be achieved only when having a dataset annotated with all needed contexts we defined. The proposed framework can be employed for general tasks, but would be outperformed by domain-specific applications.

7. Discussion

Proposed framework for matching and merging represents a general and complete solution, applicable in all diverse areas of use. Introduction of contexts allows a joint control over various dimensions of matching and merging variability, providing for specific needs of each scenario. Furthermore, data architecture combines simple (network) data with semantically enriched data, which makes the proposition applicable for any data source.

Framework can thus be used as a general solution for merging data from heterogeneous sources, and also merely for matching.

The fundamental difference between matching, including only attribute and entity resolution, and merging, including also redundancy elimination, is, besides the obvious, in the fact that merged data is read-only. Since datasets, obtained after merging, do not necessarily resemble the original datasets, the data cannot be altered thus the changes would apply also in the original datasets. Alternative approach is to merely match the given datasets and to merge them only on demand. When altering matched data, user can change the original datasets (that are in this phase still represented independently) or change the merged dataset (that was previously demanded for), in which case he must also provide an appropriate strategy, how the changes should be applied in the original datasets.

Proposed algorithms employ network data, semantically enriched with ontologies. With the advent of *Semantic Web*, ontologies are gaining importance mainly due to availability of formal ontology languages. These standardization efforts promote several notable uses of ontologies like assisting in communication between people, achieving interoperability (communication) among heterogeneous software systems and improving the design and quality of software systems. One of the most prominent applications is in the domain of semantic interoperability. While pure semantics concerns the study of meanings, semantic elevation means to achieve semantic interoperability and can be considered as a subset of information integration (including data access, aggregation, correlation and transformation). Semantic elevation of proposed matching and merging framework represents one major step towards this end.

Use of trust-aware techniques and algorithms introduces several key properties. Firstly, an adequate trust management provides means to deal with uncertain or questionable data sources, by modeling trustworthiness of each provided value appropriately. Secondly, algorithms jointly optimize not only entity resolution or redundancy elimination of provided datasets, but also the trustworthiness of the resulting datasets. The latter can substantially increase the accuracy. Thirdly, trustworthiness of data can be used also for security reasons, by seeing trustworthy values as more secure. Optimizing the trustworthiness of matching and merging thus also results in an efficient security assurance.

Although, contexts are merely a way to guide the execution of some algorithm, their definition is relatively different from that of any simple parameter. The execution is controlled with mere definition of the contexts, when in the case of parameters, it is controlled by assigning different values. For instance, when default behavior is desired, the parameters still need to be assigned, when in the case of contexts, the algorithm is used as it is. For any general solution, working with heterogeneous clients, such behavior can significantly reduce the complexity.

As different contexts are used jointly throughout matching and merging execution, they allow a collective control over various dimensions of variability. Furthermore, each execution is controlled and also characterized with the context it defines, which can be used to compare and analyze different executions or matching and merging algorithms.

Last, we briefly discuss a possible disadvantage of the proposed framework. As the framework represents a general solution, applicable in all diverse domains, the perfor-

mance of some domain-specific approach or algorithm can still be superior. However, such approaches commonly cannot be generalized and are thus inappropriate for practical (general) use.

8. Conclusion

This paper advances previously published paper (Šubelj *et al.*, 2011) which contains only theoretical view of the proposed framework for data matching and merging. In this work we again overview the whole framework with minor changes, but most importantly we introduce different metrics implementation details and full framework demonstration.

The proposed framework follows a three level architecture using network-based data representation from data to semantic and lastly to abstract level. Data on each level is always a superset of lower ones due to inclusion of various context types, trust values or additional metadata. We also identify three main context types – user, data and trust context type – which are a formal representation of all possible operations. One of the novelties is also trust management that is available across all steps during the execution.

To support our framework proposal, we conduct experiments of three main components – attribute resolution, entity resolution and redundancy elimination – using trust and semantics. Like we theoretically anticipated, results on five datasets show that semantic elevation and proper trust management significantly improve overall results.

In further work we will additionally incorporate network analysis techniques such as community detection (Šubelj and Bajec, 2011b) or recent research on self-similar networks (Blagus *et al.*, 2012), which finds network hierarchies with a number of common properties that may also improve the results of proposed approach. Furthermore, ontology-based information extraction techniques will be employed into entity resolution algorithm to gain more knowledge about non-atomic values.

References

Adamic, L., Adar, E. (2001). Friends and neighbors on the web. *Social Networks*, **25**, 211–230.

Ananthakrishna, R., Chaudhuri, S., Ganti, V. (2002). Eliminating fuzzy duplicates in data warehouses. In: *Proceedings of the International Conference on Very Large Data Bases*, pp. 586–597.

Bengtson, E., Roth, D. (2008). Understanding the value of features for coreference resolution. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 294–303.

Bhattacharya, I., Getoor, L. (2004). Iterative record linkage for cleaning and integration. In: *Proceedings of the ACM SIGKDD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 11–18.

Bhattacharya, I., Getoor, L. (2007). Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1, 5–40.

Blagus, N., Šubelj, L., Bajec, M. (2012). Self-similar scaling of density in complex real-world networks. *Physica A* (in press).

Blaze, M., Feigenbaum, J., Lacy, J. (1996). Decentralized trust management. In: *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 164–173.

Castano, S., Ferrara, A., Montanelli, S. (2006). Matching ontologies in open networked systems: techniques and applications. *Journal on Data Semantics*, 25–63.

Castano, S., Ferrara, A., Montanelli, S. (2009). The iCoord knowledge model for P2P semantic coordination. In: *Proceedings of the Conference on Italian Chapter of AIS*, pp. 229–236.

Castano, S., Ferrara, A., Montanelli, S. (2010). Dealing with matching variability of semantic web data using contexts. In: *Proceedings of the International Conference on Advanced Information Systems Engineering*, pp. 194–208.

Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., Keinberg, J. (1998). Automatic resource compilation by analyzing hyperlink structure and associated text. In: *Proceedings of the International World Wide Web Conference*, pp. 65–74.

Cohen, W.W (2000). Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18, 288–321.

Cohen, W.W, Ravikumar, P., Fienberg, S.E. (2003). A comparison of string distance metrics for name-matching tasks. In: *Proceedings of the IJCAI Workshop on Information Integration on the Web*, pp. 73–78.

Domingos, P., Richardson, M. (2001). Mining the network value of customers. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 57–66.

Dong, X., Halevy, A., Madhavan, J. (2005). Reference reconciliation in complex information spaces. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 85–96.

Euzenat, J., Shvaiko, P. (2007). *Ontology Matching*. Springer, Berlin.

Gruber, T.R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, 199–220.

Hernandez, M., Stolfo, S. (1995). The merge/purge problem for large databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 127–138.

Horrocks, I., Sattler, U. (2001). Ontology Reasoning in the SHOQ(D) Description Logic. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*.

Jaro, M.A. (1989). Advances in record linking methodolg as applied to the 1985 census of tampa florida. *Journal of the American Statistical Society*, 84, 414–420.

Joachims, T. (1997). A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In: *Proceedings of the International Conference on Machine Learning*, pp. 143–151.

Kalashnikov, D., Mehrotra, S. (2006). Domain-independent data cleaning via anlysis of entity-relationship graph. *ACM Transactions on Database Systems*, 31, 716–767.

Kautz, H., Selman, B., Shah, M. (1997). Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40, 63–65.

Kleinberg, J.M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46, 604–632.

Köpcke, H., Thor, A., Rahm, E. (2010). Evaluation of entity resolution approaches on real-world match problems. In: *Proceedings of the International Conference on Very Large Data Bases*, pp. 484–493.

Kumar, V. (1992). Algorithms for constraint-satisfaction problems: a survey. *AI Magazine*, 13, 32–44.

Lapouchnian, A., Mylopoulos, J. (2009). Modeling domain variability in requirements engineering with contexts. In: *Proceedings of the International Conference on Conceptual Modeling*, pp. 115–130.

Lavbič, D., Vasilecas, O., Rupnik, R. (2010). Ontology-based multi-agent system to support business users and management. *Technological and Economic Development of Economy*, 16(2).

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M., Jurafsky, D. (2011). Stanford's multi-pass Sieve coreference resolution system at the CoNLL-2011 shared task. In: *Proceedings of the CoNLL-2011 Shared Task*.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In: *Proceedings of the ACM SIGMOD Symposium on Principles of Database Systems*, pp. 233–246.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10, 707–710.

Lu, Q., and Getoor, L. (2003). Link-based text classification. In: *Proceedings of the IJCAI Workshop on Text*

Mining and Link Analysis, pp. 1–8.

Miller, G.A. (1995). Wordnet: a lexical database for English. *Communications of the ACM*, 38, 39–41.

Monge, A., Elkan, C. (1996). The field matching problem: algorithms and applications. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 267–270.

Moreau, E., Yvon, F., Cappe, O. (2008). Robust similarity measures for named entities matching. In: *Proceedings of the International Conference on Computational Linguistics*, pp. 593–600.

Nagy, M., Vargas-Vera, M., Motta, E. (2008). Managing conflicting beliefs with fuzzy trust on the semantic web. In: *Proceedings of the Mexican International Conference on Advances in Artificial Intelligence*, pp. 827–837.

Newman, M.E.J. (2010). *Networks: An Introduction*. Oxford University Press, London.

Ng, V. (2008). Unsupervised models for coreference resolution. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Pennerath, F., Napoli, A. (2009). The model of most informative patterns and its application to knowledge extraction from graph databases. In: *Proceedings of the International Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pp. 205–220.

Rahm, E., Bernstein, P.A. (2001). A survey of approaches to automatic schema matching. *Journal on Very Large Data Bases*, 10, 334–350.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In: *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pp. 175–186.

Richardson, M., Agrawal, R., Domingos, P. (2003). Trust management for the semantic web. In: *Proceedings of the International Semantic Web Conference*, pp. 351–368.

Roth, D., Samdani, R. (2009). Learning multi-linear representations of distributions for efficient inference. *Machine Learning*, 76, 195–209.

Sen, P., Getoor, L. (2006). Empirical comparison of approximate inference algorithms for networked data. In: *Proceedings of the International Conference on Machine Learning*, pp. 1–8.

Štajner, T., Mladenič, D. (2009). Entity resolution in texts using statistical learning and ontologies. *The Semantic Web*, 91–104.

Šubelj, L, and Bajec, M. (2011a). Robust network community detection using balanced propagation. *The European Physical Journal B*, 81(3).

Šubelj, L. Bajec, M. (2011b). Community structure of complex software systems: analysis and applications. *Physica A*, 390(16).

Šubelj, L., Jelenc, D., Zupančič, E., Lavbič, D., Trček, D., Krisper, M., Bajec, M. (2011). Merging data sources based on semantics, contexts and trust. *IPSI BgD Transactions on Internet Research*, 7(1), 18–30

Thor, A., and Rahm, E. (2007). MOMA – A mapping-based object matching system. In: *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research*.

Trcek, D. (2009). A formal apparatus for modeling trust in computing environments. *Mathematical and Computer Modelling*, 49, 226–233.

Whang, S.E., Garcia-Molina, H. (2011). Disinformation techniques for entity resolution. Technical report.

Whang, S.E., Marmaros, D., Garcia-Molina, H. (2010). Pay-as-you-go er. Technical report.

Winkler, W.E. (1990). String comparator metrics and enhanced decision rules in the Fellegi–Sunter model of record linkage. In: *Proceedings of the Section on Survey Research Methods*, pp. 354–359.

S. Žitnik is a junior researcher from industry at Optilab d.o.o. and PhD candidate with the Faculty of Computer and Information Science of the University of Ljubljana. His main research focus is information extraction from unstructured textual sources, more precisely a combination of entity, relation extraction and coreference resolution.

L. Šubelj is a researcher and a PhD candidate with the Faculty of Computer and Information Science of the University of Ljubljana. His research interests include analysis of complex networks, and selected fields in link analysis and data technologies.

D. Lavbič is assistant professor at University of Ljubljana, Faculty of Computer and Information Science. He graduated in 2004 and received his PhD in 2010 at Faculty of Computer and Information Science at University of Ljubljana. He is author of 12 original scientific papers and more than 25 published scientific conference contributions. His research interests are intelligent agents, multi-agent systems, knowledge management, ontologies, business rules and semantic web. His work is mainly oriented toward using semantic web technologies in business applications. He has been involved in several research and commercial projects on strategic planning, methodologies for IS development, using intelligent agents and business process automatization and management.

O. Vasilecas is full time professor at the information systems department, principal researcher and head of Information Systems Research Laboratory in Vilnius Gediminas Technical University. He is author and co-author of more than 250 research papers and 5 books in the field of information systems development. His research interests include knowledge, represented by business rule and ontology, based information systems development. He delivered lectures in 7 European universities including London, Barcelona, Athens and Ljubljana. O. Vasilecas carried out an apprenticeship in Germany, Holland, China and last time in Latvia and Slovenia universities. He supervised 9 successfully defended doctoral theses and now is supervising 5 more doctoral students. He was leader of number of international and local projects. Last time he leaded "Business Rules Solutions for Information Systems Development (VeTIS)" project carried out under High Technology Development Program of Lithuania.

M. Bajec is an associate professor at the Faculty of Computer & Information Science at the University of Ljubljana, Slovenia. His main research interests include IT Governance, specifically the management and engineering of software development methods and IT/IS strategy planning. Since 2009, he has been on the position of the head of the Laboratory for Data Technologies where he manages research in data acquisition, integration, analysis, visualisation and management. Marko Bajec has over 15 years of research experiences. His bibliography counts over 120 items, among which 30 are original scientific papers, published in peer reviewed journals. In the last 10 years he has led or coordinated over 20 domestic and international projects. In 2010, Marko Bajec was awarded from the Slovenian Society Informatika for his contribution in transferring knowledge to industry.

Bendras kontekstui jautrus duomenų susiejimo bei suliejimo karkasas

Slavko ŽITNIK, Lovro ŠUBELJ, Dejan LAVBIČ, Olegas VASILECAS, Marko BAJEC

Atsižvelgiant i tai, kad egzistuoja skaitlingas kiekis viešai prieinamų informacijos šaltinių ir paslaugų, pasiūlyta nemažai metodų skirtų tokių paprastai heterogeninių duomenų sujungimui. Tačiau bendro pobūdžio šaltinis – gavėjas sprendimai vis dar gana retai pateikiami, ypatingai tais atvejais, kai dėmesin reikia priimti skirtingų kontekstų aspektus. Be to, tokie sprendimai dažnai nepakankamai išbandyti arba pritaikyti tik specifinėms dalykinėms sritims. Straipsnyje trumpai apžvelgtas ir išsamiai įvertintas pasiūlytas bendro pobūdžio susiejimo ir suliejimo karkasas. Jis naudoja bendrąją esybių rezoliuciją ir perteklišlumo šalinimą naudojant tris kontekstų tipų aspektus. Siekiant gauti nuo dalykinės srities nepriklausomus rezultatus, analizuojami duomenys praturtinami semantikos ir pasitikėjimo atributais. Tačiau pagrindinis straipsnio indelis susijęs su penkių viešai prieinamų tarpusavyje nesuderintų domenų duomenų rinkinių įvertinimų. Be to, mes įvedėme papildoma atributą, ryšį bei semantikos ir pasitikėjimo metriką, kas įgalina užtikrinti pilną pasiūlyto karkaso valdymą. Papildomai prie to, kad naudojant karkasą pagerėjo rezultatai, pasiūlytos metrikos gali būti atskirai teikiamos kaip svarbus tyrimų rezultatas.