MOMA - A Mapping-based Object Matching System

Andreas Thor
University of Leipzig, Germany
thor@informatik.uni-leipzig.de

Erhard Rahm
University of Leipzig, Germany
rahm@informatik.uni-leipzig.de

ABSTRACT

Object matching or object consolidation is a crucial task for data integration and data cleaning. It addresses the problem of identifying object instances in data sources referring to the same real world entity. We propose a flexible framework called MOMA for mapping-based object matching. It allows the construction of match workflows combining the results of several matcher algorithms on both attribute values and contextual information. The output of a match task is an instance-level mapping that supports information fusion in P2P data integration systems and can be re-used for other match tasks. MOMA utilizes further semantic mappings of different cardinalities and provides merge and compose operators for mapping combination. We propose and evaluate several strategies for both object matching between different sources as well as for duplicate identification within a single data source.

1. MOTIVATION

Object matching (also known as object consolidation, duplicate identification, record linkage, entity resolution or reference reconciliation) is a crucial task for data integration and data cleaning [9, 18, 27] and its history goes back over 20 years [3, 19] . It addresses the problem of identifying object instances referring to the same real world entity. The instances may reside in different, typically heterogeneous data sources or may already be stored in a single data source, e.g., in a structured database or a search engine store. The instances to be consolidated may be physically materialized or dynamically be requested from sources, e.g. by database queries or keyword searches.

The high importance and difficulty of the object matching problem has triggered a huge amount of research on different variations of the problem. Most previously proposed approaches focus on matching relational records and apply what we call attribute matching, i.e., they use the values of selected attributes to determine the similarity between instances. These approaches have also been called "fuzzy join" or "fuzzy match" [6, 7] and considered different similarity functions (e.g., different types of string similarity) [10] and their efficient implementation. More recently, authors have recognized the value of considering additional information for object consolidation, in particular semantic relationships or mappings. For instance, [4] uses the co-authorship relationship to match author instances, i.e., two authors are considered as highly similar if they have the same co-authors. Several studies proposed graph matching algorithms to consider such contextual information, e.g., [4, 8, 11, 35]. The use of relationship information is especially promising when the effectiveness of attribute matching is low, e.g. for sources with only few attributes or in the presence of dirty or highly different value representations (e.g., "CIDR 2007" vs. "3rd Biennial Conference on Innovative Data Systems Research").

We propose a flexible and domain-independent framework called MOMA for ***m***apping-based ***o***bject ***ma***tching. The design of MOMA is inspired by our previous work on schema matching, especially the COMA approach [13] which allows the combined use of multiple match algorithms for a given schema match problem. MOMA also uses an extensible library of match algorithms, both attribute and context matchers, and the combination of their results to improve match quality. Of course, the focus now is on matching real, dirty data which may not have a rich schema (e.g., web data) and typically is much more voluminous than metadata / schemas.

A key feature of MOMA is that it is massively based on the notion of instance mappings. The output of a match task is represented as a so-called *same-mapping* indicating which input objects are considered semantically equivalent. Automatically generating such mappings is especially useful for peer-to-peer (P2P) data integration to map a new data source to already integrated peers. The generated mappings allow us to traverse between peers and to fuse together and enhance information on equivalent objects for data analysis and query answering.

To solve a particular match problem and generate a corresponding same-mapping MOMA allows the specification of a workflow of several steps each of which combines several existing mappings or matcher executions. Hence, already existing same-mappings and the output of previous match steps can be iteratively refined during a match workflow. MOMA's neighborhood matcher additionally utilizes so-called association mappings representing semantic relationships between different objects, such as publications of authors or publications of a conference. For instance, we can consider two conferences at different data sources as equivalent if the majority of their publications match with each other. MOMA also provides self-tuning capabilities to automatically find optimal configurations for a match task.

A first version of MOMA has been implemented based on our P2P data integration platform iFuice [30]. The use of iFuice for a large-scale citation analysis involving automatically generated web sources like Google Scholar [29] revealed the strong need for a sys-

3rd Biennial Conference on Innovative Data Systems Research (CIDR) January 7-10, 2007, Asilomar, California, USA.

tem like MOMA to achieve high-quality object matching with little or no manual data cleaning activities. In this paper, we will also use examples from the bibliographic domain for illustration.

The main contributions of this paper are the presentation of the new MOMA architecture for mapping-based object matching and the specification of several match strategies to combine match results and existing mappings. Furthermore we present results of an extensive evaluation on real data sources (including Google Scholar) indicating the high effectiveness of MOMA for both object matching between different sources as well as within a single data source.

The next section introduces some definitions and outlines the MOMA architecture. Section 3 describes the mapping combination techniques and Section 4 presents typical workflows for object matching and duplicate detection. Section 5 provides an initial evaluation on real-world bibliographic datasets. We review related work in Section 6 before we conclude in Section 7.

2. OVERVIEW OF MOMA

2.1 Definitions and Problem Statement

MOMA's model for sources and mappings follow the assumptions of the iFuice data integration platform [30]. We differentiate between physical data sources (e.g., DBLP, ACM Digital Library or Google Scholar) and logical data sources. Each logical data source (LDS) belongs to one physical data source (PDS) and consists of object instances of a particular semantic object type (e.g., Publication or Author). Each object instance is identified by an id value and may have additional attribute values. Figure 1 shows some object instances from the sources DBLP and ACM.

Mappings interconnect LDS and have a specific semantic mapping type, e.g. publications of author. We focus on instance mappings which consist of correspondences between two LDS. PDS, LDS and mappings are represented in a so-called source-mapping model (SMM) as illustrated in Figure 2.

Definition 1 (Mapping and correspondence): A mapping m between LDS_A and LDS_B consists of a set of correspondences { (a, b, s) | $a \in LDS_A$, $b \in LDS_B$, $s \in [0,1]$ }. The similarity value s indicates the similarity or strength of the correspondence between a domain object $a \in LDS_A$, and a range object $b \in LDS_B$.

Like [20] we represent mappings by a mapping table with three columns. Each row represents a correspondence consisting of the ids of the domain and range objects and the corresponding similarity value.

We distinguish between same-mappings and association mappings. *Same-mappings* connect instances of the same object type and represent a semantic equality relationship. Figure 1 illustrates an example of a same-mapping for publications. The shown similarity values could be a result of combining two attribute matchers on publication title and year. All mappings that are not same-mappings are called *association mappings*, e.g. of type publication-author or publication-venue. The publication-venue mapping is of cardinality n:1 so that we could have simply represented the venue information as an attribute per publication object. The advantage of introducing the object type *venue* and an explicit mapping representation is twofold. First, we can easily determine and use the inverse mapping, i.e., to get all publications for venues. Second, we gain flexibility for mapping combination. For example, we can compose the publication-author and publication-venue mappings to determine all authors publishing at a certain venue. As we will see we can also utilize these association mappings to determine same-mappings.

The SMM of Figure 2 only shows association mappings for simplicity. Since same-mappings can be established between all LDS of the same object type, there may be up to 8 same-mappings (3 for publications, 3 for authors, 2 for venues). The goal of object matching is to identify the correspondences for such same-mappings.

Definition 2 (Problem statement: object matching): Object matching is a process which takes as input two sets of objects $A \subseteq LDS_A$ and $B \subseteq LDS_B$ of the same object type. The output is a same-mapping between A and B containing correspondences between objects of A and B representing the same real-world entity. The goal is to achieve a high-quality mapping with respect to recall and precision, i.e. all real correspondences but no other object pairs should be included in the result.

Note that the inputs need not be entire LDS but only subsets of LDS. This is because web sources like Google Scholar do not support downloading all their data but only support querying selected subsets. Hence, object matching needs to be performed on the results of such queries. Furthermore, we are aiming at supporting both extensive offline matching of large data sets (e.g. for physical integration such as data warehousing) and small-sized online matching (e.g. during query processing in virtual data integration scenarios).

MOMA also supports duplicate detection within a single source by matching instances of the same LDS with each other. We refer to the resulting same-mappings as *self-mappings*; they represent all duplicate records within the source. Google scholar uses sophisticated clustering to determine matching documents and corresponding citations. However their matching is not perfect and can lead to wrong (citation) analysis results without additional object matching. One goal of MOMA is to improve the duplicate identification within such sources by utilizing additional matchers and mappings.

Pubclication@DBLP

Key	Title	Pages	Year
conf/VLDB/MadhavanBR01	Generic Schema Matching with Cupid	49-58	2001
conf/VLDB/ChirkovaHS01	A formal perspective on the view ...	59-68	2001
journals/VLDB/ChirkovaHS02	A formal perspective on the view ...	216-237	2002

Pubclication@ACM

Id	Name	Citations	Year
P-672191	Generic Schema Matching with Cupid	69	2001
P-672216	A formal perspective on the view ...	10	2001
P-641272	A formal perspective on the view ...	1	2002

Same-mapping

Publication@DBLP	Publication@ACM	Sim
conf/VLDB/MadhavanBR01	P-672191	1
conf/VLDB/ChirkovaHS01	P-672216	1
conf/VLDB/ChirkovaHS01	P-641272	0.6
journals/VLDB/ChirkovaHS02	P-672216	0.6
journals/VLDB/ChirkovaHS02	P-641272	1

Figure 1. Examples for object instances (left and middle) and same-mapping (right)

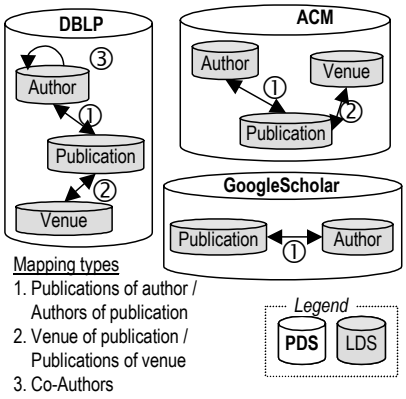


Figure 2. Source mapping model for bibliographic domain

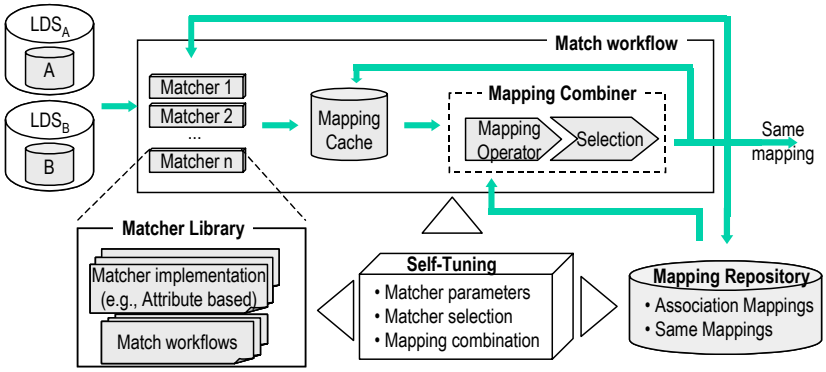


Figure 3. Schematic overview of MOMA and match workflows

2.2 MOMA Architecture and Match Workflow

Figure 3 illustrates components of the MOMA architecture as well as the process of object matching. A mapping repository is used to materialize both association and same-mappings. Given the simple structure of our mappings they can efficiently be maintained in relational mapping tables. Many mappings already exist in data sources and can thus be utilized for object matching. For instance, DBLP data on publication lists for venues and for authors are kept as association mappings. Similarly, some same-mappings exist already, e.g., Google Scholar links its publications to ACM. MOMA also maintains a mapping cache for storing intermediate same-mappings derived during a match workflow. Hence, MOMA not only processes the input instances but also utilizes the mappings of the repository and the cache for object matching.

There is an extensible library of matcher algorithms that can be used for a specific match task. Matchers conform to the same interfaces as a match process, in particular they generate a same-mapping. Otherwise there is no restriction on the implementation of matchers, e.g., they can be attribute matchers or context matchers like graph-based matching algorithms (e.g., [11]). Furthermore, they may utilize a variety of similarity computations, e.g. based on string matching, use of auxiliary information like dictionaries or use of query functionality to access data sources. Certainly powerful text matching algorithms are especially important for web objects / documents. In our current implementation, we use a generic attribute matcher that is provided with a pair of attributes to be matched, a similarity function to be evaluated (e.g. n-gram, TF/IDF or affix) and a similarity threshold to be exceeded by result correspondences. A multi-attribute matcher is also supported which directly evaluates and combines the similarity for multiple attribute pairs, e.g., for publication title and publication year.

The MOMA match process is a workflow consisting of a sequence of steps. Each such step generates a same-mapping that can be refined by additional steps. Selected workflows can be added to the matcher library for use in other match tasks. The final same-mapping determined by a match process is stored in the mapping repository and can be re-used in other workflows.

Each workflow step consists of two parts: matcher execution and mapping combination. The execution of selected matchers is actually optional, i.e., a step may only combine existing or previously computed mappings from the mapping repository or mapping cache. The combination of mappings in a step is processed by a specific *mapping combiner*. The input of a mapping combiner is a list of mappings, either from the mapping cache or mapping repository, the output is a same-mapping. A combiner is specified by a mapping operator followed by an optional selection. The mapping operator specifies how the resulting correspondences are determined from the input mappings, e.g. by a merge or compose operation (see Section 3). The *selection step* filters the correspondences to restrict the mapping to the most similar instances. Selections are typically specified by constraints on the similarity values, e.g., a *Threshold* constraint returns all correspondences above a given similarity value. Additionally the selection step can consider domain-specific constraints, e.g., to require that the publication year of matching publications should not differ by more than one year.

MOMA provides a high flexibility to determine a tailored workflow for a given match task. In particular, it allows selection and combination of several matchers and the re-use and refinement of previously determined mappings. Several match strategies will be described in more detail in the next section. On the other hand, the high flexibility can make it difficult even for experts to specify a suitable workflow and configuration. Similar to the E-Tuner approach for schema matching [31], MOMA therefore will provide self-tuning capabilities to automatically select matchers and mappings and to find optimal configuration parameters. Initially the focus is on optimizing individual matchers and combination schemes. For example, for attribute matching choices must be made on which attributes to match, and which similarity function and similarity threshold to apply. For suitable training data these parameters can be optimized by standard machine learning schemes, e.g. using decision trees.

3. Mapping Operators

We propose two types of mapping combination namely merging and composition. Merging is applicable for mappings between two LDS of the same object type, LDS_A and LDS_B, and determines a

map1		
a1	b1	1
a2	b2	0.8

map2		
a1	b1	0.6
a1	b5	1
a3	b3	0.9

(a) Input mappings

merge (Min-0)		
a1	b1	0.6

merge (Avg)		
a1	b1	0.8
a1	b5	1
a2	b2	0.8
a3	b3	0.9

merge (Avg-0)		
a1	b1	0.8
a1	b5	0.5
a2	b2	0.4
a3	b3	0.45

merge (Prefer map1)		
a1	b1	1
a2	b2	0.8
a3	b3	0.9

(b) Merge results for different similarity functions

Figure 4. Example execution of merge operator

combined mapping between these sources. Mapping composition aims at deriving a mapping from LDS_A to LDS_B with the help of two mappings from LDS_A to LDS_C and from LDS_C to LDS_B. Note that the operators need to process and generate "fuzzy" mappings, i.e., we need methods to determine the similarity values for the correspondences in the output mappings. In MOMA merging is only needed for combining same-mappings while compose is used for both same-mappings and association mappings.

In the following we define the operators for merging and composing mappings. Afterwards we briefly present our selection operator.

3.1 Merging Mappings

Our framework supports a general n-ary merge operator to unify the correspondences of mappings of the same type. Given *n* mappings map_i between LDS_A and LDS_B, the merge operator returns a combined mapping between these sources. An additional input parameter for merge is a combination function *f*. It determines how the similarity values s_i of input correspondences *(a,b,s_i)* should be combined for the merged correspondence *(a,b,s)*. We currently provide the following functions:

- *Avg* / *Min* / *Max*: average / minimum / maximum value of the input similarities s_i
- *Weighted*: Weighted average
- *PreferMap_i*: This approach prefers one of the input mappings, e.g., if it is known to provide good quality.

For *Min*, *Average* and *Weighted Average* we can also specify how to deal with correspondences that are missing in some of the input mappings. The default strategy ignores such missing correspondences and only considers the available similarity values. This approach is especially useful for input mappings containing correspondences only for a small fraction of the objects. Such incomplete mappings are quite common when sources overlap only partially or have many missing values (e.g. for optional attributes such as publication year in Google Scholar). By ignoring missing correspondences incomplete mappings can contribute to the combined mapping result (and thus improving recall) without reducing the similarity values for correspondences of other input mappings.

On the other hand we may choose to assume a similarity value of 0 for a missing correspondence in order to improve precision of the result mapping. Figure 4 illustrates the effect of different similarity functions *f* for two input mappings. Min-0 and Avg-0 refer to the variants where missing similarity values are assumed to be 0; Avg only considers existing correspondences. Note that *Min-0* has the semantics of a mapping *intersection* filtering away all correspondences which are not present in all input mappings. Such a restrictive merge approach is expected to improve the precision of same-mappings at the expense of a reduced recall.

The *PreferMap* similarity function aims at providing both good recall and precision by preferring one mapping which is known to provide good quality. It includes all correspondences from the preferred mapping and adds only such correspondences from the other mapping(s) for domain instances that do not have any correspondence in the preferred mapping. In the example of Figure 4, the merged mapping contains all correspondences of the preferred mapping *map1*. Since *map1* already „covers" domain instances *a1* and *a2* only the correspondence (*a3*,*b3*) from *map2* is added to the result mapping. The intuition behind this is that the non-preferred mappings should only contribute non-conflicting matches for otherwise uncovered objects (thus improving recall) but not reduce the precision for the correspondences of the preferred mapping.

3.2 Mapping Composition

Another combination strategy involves the composition of mappings. Assume we have a mapping map_1 from LDS_A to LDS_C and a second mapping map_2 from LDS_C to LDS_B. The composition of these two mappings will relate LDS_A and LDS_B.

The compose result contains all correspondences *(a,b,s)* for which exists at least one object c_i so that *(a,c_i,s_i1)* and *(c_i,b,s_i2)* occur in the initial mappings. The computation of the similarity value *s* of the output correspondences is somewhat more complex than for the merge operator. This is because a composed correspondence may be reached via different compose paths, and each compose path with two mappings provides two similarity values, s_{i1} and s_{i2}. We use a combination function *f* to first combine the two confidence values s_{i1} and s_{i2} per compose path to a combined value s_i. For *f* we have the same alternatives as for the merge operator. A second combination function *g* is used to aggregate the similarity values s_i for all compose paths *(a,c_i,b)* for the resulting correspondence *(a,b,s)*. For *g* we support the following alternatives (see Figure 5 for the definition of *n(a)*, *n(b)*, and *s(a,b)*):

- *Avg* / *Min* / *Max*: Average, minimum, maximum value of all compose path similarity values s_i

$$n(a) = \left|\left\{c_i \middle| \exists s_i : (a, c_i, s_i) \in map_1\right\}\right|$$

$$n(b) = \left|\left\{c_i \middle| \exists s_i : (c_i, b, s_i) \in map_2\right\}\right|$$

$$s(a,b) = \sum \left\{ f(s_{i1}, s_{i2}) \middle| \exists c_i : (a, c_i, s_{i1}) \in map_1 \wedge (c_i, b, s_{i2}) \in map_2 \right\}$$

Figure 5. Auxillary values for *Relative* similarity functions

map1		
v1	p1	1
v1	p2	1
v1	p3	0.6
v2	p2	0.6
v2	p3	1

map2		
p1	v'1	1
p2	v'1	1
p3	v'2	1

compose		
v1	v'1	0.8 = 2 * (1+1) / (3+2)
v1	v'2	0.3 = 2 * 0.6 / (3+1)
v2	v'1	0.3 = 2 * 0.6 / (2+2)
v2	v'2	0.67 = 2 * 1 / (2+1)

Figure 6. Example execution of compose operator using the similarity functions f = Min and g = Relative

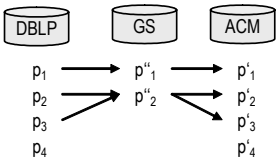


Figure 7. Composing same-mappings

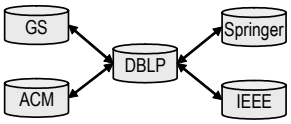


Figure 8. Hub infrastructure-for composing same-mappings

- *RelativeLeft* = $s(a,b) / n(a)$ = ratio of the sum of all compose path similarity values s_i and the number of correspondences for object *a* in the first (left) input mapping map_1
- *RelativeRight* = $s(a,b) / n(b)$
- *Relative* = $2 \cdot s(a,b) / (n(a)+n(b))$ = harmonic mean of *RelativeLeft* and *RelativeRight*, i.e., twice the sum of all s_i values divided by the sum of correspondences for a and b in map_1 and map_2, respectively.

The *Relative* approaches consider the number of compose paths to prefer correspondences that are reached via multiple paths. This is illustrated by the composition example of Figure 6 using the *f=Min* and *g=Relative* functions. The goal is to derive a venue same-mapping by composing a venue-publication with a publication-venue mapping. Consider the result correspondence between venues v1 and v'1 which can be reached via two publications p1 and p2. Both compose paths contribute with their path similarity of min(1,1)=1. The *Relative* function divides twice the sum of all path similarity values, i.e. 2*2=4, by the number of correspondences for v1 in map1 (=3) and the number of correspondences of v'1 in map2 (=2). Hence, we obtain a final similarity value of s= 4/(3+2)= 0.8. The example illustrates that the *Relative* similarity function takes into account the number of compose paths as well as the sum of the compose path similarity values. In the example, the output correspondence (v1,v'1) receives a higher similarity value than (v1, v'2) since it is supported by more compose paths (2 matching publications vs. 1).

3.3 Selection of Correspondences

Selection is the second step of a mapping combiner MC to eliminate less likely correspondences from a same-mapping. MOMA supports the following techniques:

- *Threshold*: All correspondences above a given similarity value are selected.
- *Best-n*: The n correspondences having the highest similarity value are selected for each domain (range) instance.
- *Best-1+Delta*: The correspondence with maximal similarity value is determined for all domain (range) instances plus all correspondences with a similarity differing at most by a tolerance value *d*. The delta d can be specified either as an absolute or relative value. The idea is to return multiple results when there are several correspondences with almost the same top similarity value.
- *Object value constraint*: Only correspondences that fulfil a certain domain-specific constraint are considered. For example, a publication same-mapping may have to satisfy the constraint that the publication years of matching objects must not differ by more than one year.

4. MATCH STRATEGIES

The MOMA framework has been implemented within the iFuice data integration platform. In addition to the introduced mapping operators, iFuice supports other operators for querying data sources, accessing object instances based on their ids, traversing mappings, and aggregating objects interconnected by same-mappings [30]. Same-mappings generated by the MOMA approach can thus be used in iFuice applications for aggregating (or fusing) information from different sources. For example, DBLP publications can be combined with their matching publications in ACM DL and Google Scholar to obtain additional attribute values like the number of citations or author institutions. In iFuice, operators can be used within script programs to perform data access and mapping execution. We use this script facility to implement our match workflows based on the introduced mapping operators.

To illustrate the power of the MOMA framework we outline sample match workflows (or workflow steps) in this section. We first discuss the combination of several same-mappings based on the merge and compose operators. We then discuss the utilization of association mappings within a so-called neighborhood matcher. We will see that the quality of neighborhood matcher depends on the cardinality of the association mappings and propose different match workflows to deal with them. Finally, we consider the special case of duplicate detection within a single data source.

4.1 Combination of Same-Mappings

4.1.1 Independently executed matchers

The simplest form of a match workflow is the execution of n independent matchers on the two input sources followed by merging the n same-mappings. Merging aims at complementing the results of individual matchers and overcoming shortcomings of different matchers for certain instances. For example, we may execute several attribute matchers on one or several attributes. In the bibliographic application, the DBLP and ACM publications can be match based on their titles and years. Merging both same-mappings leads to a combined match result (e.g., as in Figure 1).

4.1.2 Compose Paths of Same-Mappings

A promising way to generate a same-mapping is the composition of existing same-mappings thereby re-using these mappings. Since same-mappings conceptually represent 1:1 mappings their composition should also result into 1:1 mappings, i.e., the composition of same-mappings should be transitive. In the simplest case, only two

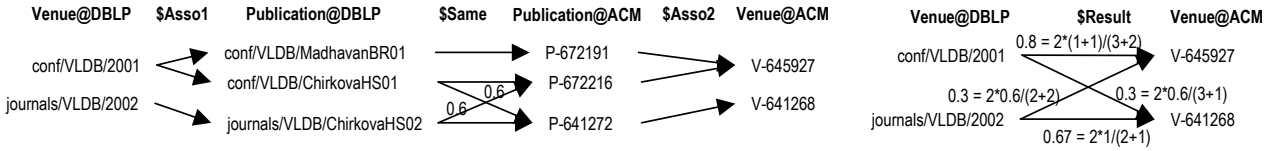


Figure 9. Sample execution of neighborhood matcher for DBLP venues based on publication same-mapping

same-mappings are composed. For example, a same-mapping map_3 between LDS_A and LDS_B can be determined by composing two same-mappings map_1 between LDS_A and LDS_C and map_2 between LDS_C and LDS_B. Similarly, more than two mappings can be composed to derive a desired same-mapping. Furthermore, there may be several applicable compose paths so that their results may be merged to derive a combined same-mapping.

The quality of a compose result is largely influenced by the cleanliness and coverage of the sources involved. For example, same-mappings may relate an instance to several object instances (duplicates) at another source, e.g. a DBLP publication may have several corresponding entries at Google Scholar (GS) or vice versa. The coverage of data sources may be vastly different so that many instances may not have a counterpart in another data source. Fig. 7 illustrates how these factors can impair match quality when composing two publication same-mappings between DBLP-GS and GS-ACM. Publications p_2 and p_3 are assumed to have the same title, e.g., a conference and a journal version of a paper. DBLP and ACM correctly differentiate these two publications while the same-mappings with GS are assumed to map them to a single publication object. Hence composing the two same-mappings would not cleanly match the DBLP with the ACM publications (4 instead of 2 correspondences) so that match precision is impaired. Even worse, the correspondence between p4 and p'4 cannot be derived by the composition (thus lowering recall) since they have no corresponding object in the intermediate source GS.

We propose two general ways to deal with such problems. First, combining an imperfect compose result with the results of other compose paths and other matchers, e.g. attribute matchers, is likely to help in many cases. Secondly, the intermediate sources within compose paths should be carefully selected and provide high quality with respect to their coverage and cleanliness (lack of duplicates). In the bibliographic domain for Computer Science the manually curated DBLP source is a good candidate for an intermediate source or even a *hub source* to which many other sources should be connected with a same-mapping (see Fig. 8). All data sources connected with the hub can efficiently be matched with each other. Generating a same-mapping between any two sources only requires the composition of two same-mappings via the hub.

4.2 Utilizing Association Mappings

The generation of same-mappings can also be achieved by utilizing association mappings with other data sources for which a same-mapping has already been established. For example, assume we have already a same-mapping for publications between DBLP and ACM as well as association mappings of type publications of venue. Starting from a DBLP venue we can then traverse to the associated DBLP publications, utilize the same-mapping to find the corresponding ACM publications, and traverse the association mapping to reach the ACM venue. We can thus generate a venue same-mapping by composing the association mappings with the publication same-mapping.

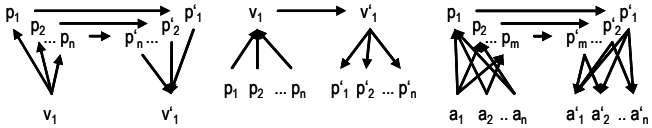
We generalize this idea by proposing the so-called neighborhood matcher using the iFuice script language [30]:

```
1: PROCEDURE nhMatch ( $Asso1, $Same, $Asso2)
2:    $Temp = compose ( $Asso1 , $Same , Min, Average )
3:    $Result = compose ( $Temp , $Asso2 , Min, Relative )
4:    RETURN $Result
5: END
```

The inputs are the two association mappings of inverse semantic mapping type (e.g., VenuePub and PubVenue) and a same-mapping. The matcher is simply a sequence of two compose operations. We start with composing the first association mapping and the same-mapping. In a second step we compose the resulting mapping with the second association mapping. For the second composition we use the similarity function *Relative* to prefer correspondences reached via multiple compose paths.

Figure 9 gives a concrete example for the execution of the neighborhood matcher using the same-mapping of Fig. 1. The first compose operator generates a mapping from Venue@DBLP to Publication@ACM. The execution of the second compose operator is already illustrated in Figure 6 and leads to a venue same-mapping. Although both DBLP venues have correspondences to both considered ACM venues, the similarity values indicate the confidence of such correspondences. The *Relative* combination function thus prefers correspondences reached via multiple compose paths and reduces the influence of wrong correspondences in the underlying publication same-mapping. Therefore a threshold-based selection could be applied afterwards to determine the desired venue mapping.

The potential usefulness of the neighborhood matcher depends on the cardinality of the utilized association mapping: 1:n, n:1, and



(a) 1:n (b) n:1 (c) n:m

(Venue-Publication) (Publication-Venue) (Author-Publication)

Figure 10. Neighborhood matching w.r.t.semantic cardinality

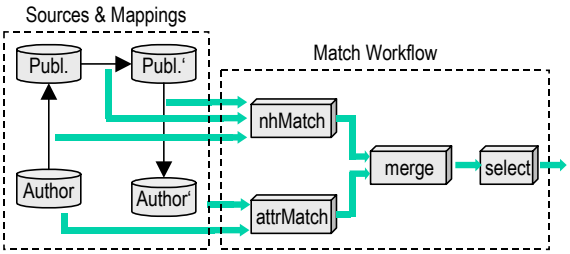


Figure 11. Example match workflow for n:m case

n:m. Figure 10 shows examples for these three cases. The previously discussed example used the mapping from venues to publications and thus illustrates the 1:n case (see also Figure 10a). We can expect good match results in this case because each venue is likely to be mapped to only one (or few) corresponding venue(s) at the other source.

While the prospects are less positive in the n:1 and n:m cases the neighborhood matcher is still promising since it can restrict the number of match candidates. In the n:1 example of Figure 10b publications cannot perfectly be matched but the resulting correspondences refer to publications published at the same venue so that the match candidates are confined compared to all publications. Even in the n:m case the search space for match candidates can be reduced. The example in Figure 10c illustrates this case for the association mapping author-publication, i.e., we want to derive an author same-mapping by utilizing an existing publication same-mapping. The result of the neighborhood matcher returns a correspondence between two authors if they share at least one publication. Due to the combination function Relative author correspondences reached via multiple paths (publications) achieve a higher similarity value.

The examples show that the neighborhood matcher alone is insufficient to determine a useful same-mapping for the n:1 and n:m cases. However the result is still useful as it can be refined by applying an additional matching, e.g. by using an attribute match on author name (see Figure 11). These matchers can work on rather small input data. As our evaluation has showed this helps to significantly improve match quality compared to the usage of attribute matching on the complete input data sets.

4.3 Duplicate Detection with Self-Mappings

The MOMA framework can also be used to determine self-mappings, i.e. same-mappings within one LDS. We observe that all previously discussed match strategies can also be applied for this special case to detect duplicates within a source. In particular, we can combine multiple attribute matchers on one source and re-use existing same-mappings and association mappings by composing them in a useful order.

For example, the study in [23] focused on identifying duplicate authors based on the co-authorship relationship. The underlying assumption is that two authors are likely duplicates if they share a significant number of co-authors. In our framework such an approach corresponds to the n:m case for the neighborhood matcher with an identity mapping on authors as a trivial same-mapping.

Table 1. Number of instances for the considered data sources

	Venues	Publications	Authors
DBLP	130	2.616	3.319
ACM DL	128	2.294	3.547
Google Scholar	-	64.263	(81.296)

The following script implements such a match strategy to detect duplicate authors in DBLP. The first line uses the neighborhood matcher on the existing DBLP co-authorship mapping to generate a same-mapping between authors. The similarity values indicate the overlap of the co-author lists. Since the neighborhood matcher is executed within only one data source, the second parameter is an identity mapping of DBLP authors. The second line generates an author same-mapping based on name similarity of authors that is computed by the trigram metric. Both mappings are merged (line 3) to identify duplicate candidates. The final selection step eliminates "trivial duplicates", i.e., correspondences between equal authors.

```
1: $CoAuthSim = nhMatch (DBLP.CoAuthor, DBLP.AuthorAuthor,
                         DBLP.CoAuthor)
2: $NameSim = attrMatch (DBLP.Author, DBLP.Author,
                         Trigram, 0.5, "[name]", "[name]")
3: $Merged = merge ($CoAuthSim, $NameSim, Average)
4: $Result = select ($Merged, "[domain.id]<>[range.id]")
```

Duplicate detection is especially important for automatically generated data collections like Google Scholar. This source clusters already publications together, presumably based on attribute values like paper title and author names. In our framework, we can represent these clusters as a self-mapping and use additional matchers to refine it for improved duplicate detection. For example, we may improve the existing self-mapping by composing it with a publication same-mapping to DBLP.

5. EVALUATION

To demonstrate the practicality and usefulness of the MOMA framework we have applied it for object consolidation on comparatively large bibliographic data sets. After describing the evaluation setup we describe the effectiveness of various match strategies introduced in Section 4, in particular merging several same-mappings (5.2), composing same-mappings (), neighborhood matching (5.4) and duplicate detection (5.5). Finally we summarize the main observations from the evaluation (5.6).

5.1 Evaluation Setting

Our evaluation refers to three physical data sources already considered so far: the DBLP bibliography[1], ACM Digital Library[2] and Google Scholar[3]. Each of them provides information about publications, authors and venues (see source-mapping model of Fig. 2) but they are very different in terms of data quality, coverage and accessibility. DBLP is manually curated and of high quality. It can be completely downloaded, and we store it in a relational database.

1.http://www.informatik.uni-trier.de/~ley/db/index.html
2.http://portal.acm.org/portal.cfm
3.http://scholar.google.com

Table 2. Matching DBLP-ACM publications using attribute matchers

	Title	Author	Year	Merge
Precision	86.7%	38.0%	0.4%	97.3%
Recall	97.7%	87.9%	100%	93.9%
F-Measure	91.9%	53.1%	0.8%	95.5%

Table 3. Matching publications via different compose paths (F-Measure)

Matcher	DBLP - GS (Compose via ACM)	DBLP - ACM (Compose via GS)	GS - ACM (Compose via DBLP)
Direct	81.3%	91.9%	35.3%
Compose	33.9%	63.7%	83.9%
Merge	81.3%	91.6%	83.7%

ACM Digital Library and Google scholar are web data sources which cannot be downloaded. They can both be accessed by queries. Like DBLP, ACM DL also provides complete lists of publications per venue. Google Scholar covers a huge number of documents automatically crawled from the web but also includes the papers from digital libraries including ACMDL. GS automatically extracts the bibliographic data from the reference sections of the documents which may lead to quality problems.

Our evaluation considers database publications from 1994 to 2003 which appeared in the conferences VLDB and SIGMOD as well as in the journals TODS, VLDB Journal and SIGMOD Record. A comprehensive citation analysis based on this data can be found in [29]. Table 1 shows the number of involved venues, authors, and publications. For Google Scholar we had to send numerous queries for generating the relevant Google Scholar references. Those queries contain the publication titles as well as venue names from the considered DBLP publications. While our datasets may seem small they are significantly larger than the ones used in previous object consolidation evaluations. For example, [11] utilizes a single dataset, Cora, consisting of 6.107 bibliographic references for only 338 publications. Furthermore, we consider three real data sources with rather different characteristics.

The goal for object consolidation in the considered setting is the automatic determination of same-mappings for publications between the three sources and for authors and venues between DBLP and ACM. We measure the quality of different match workflows with the standard metrics precision, recall and F-measure with respect to manually determined „perfect“ mappings. These manually confirmed mappings were also used in the citation analysis [29].

5.2 Merging Same-Mappings

In our first experiment we use three attribute matchers to determine a publication same-mapping between DBLP and ACM. The first two matchers perform string (trigram) matching on the publication titles and author names, whereas the last matchers compares publication years. Table 2 shows the precision, recall and F-measure results obtained for these matchers as well as for their combinations using the merge operator (using the Avg function and 80% threshold selection).

We observe that the title matcher outperforms the other matchers and achieves already a rather high match quality leaving comparatively little room for improvement. Still we can increase match quality by merging multiple same-mappings thus confirming the usefulness of combining matchers. We will also show the benefit of merging mappings in the following sections where we utilize the merge operators within other match strategies.

5.3 Composing Same-Mapping

In this experiment we want to analyze the usefulness of composing existing same-mappings to derive new same-mappings. Reusing such mappings is also attractive since the composition can be computed very efficiently in our implementation by joining the mapping tables. We consider the generation of publication same-mappings between all three sources. For any two sources we determine a direct same-mapping. We utilize the string-based attribute matcher on publication title for mappings between DBLP-ACM and DBLP-GS. For the same-mapping between GS and ACM we utilize an existing mapping by extracting existing links in the GS publication entries linking to ACM. In Table 3 we compare the quality of these direct mappings with the composed mapping via the third data source.

We observe that in one of the three cases, GS-ACM, the composed mapping is much more effective than the direct match result. This is influenced by a poor recall (21.6%) of the existing GS-ACM links. The poor GS-ACM mapping also deteriorated the quality of the compose paths DBLP-GS-ACM and DBLP-ACM-GS making them much less effective than applying a direct attribute matching. The compose path DBLP-ACM-GS also suffered from the fact that ACM DL is incomplete since it misses all publications for VLDB 2002/2003. The results show that the compose approach is very dependent on the chosen compose path and that the intermediary source should be of high quality such as DBLP.

Users may not always know the quality of the available sources and mappings. Fortunately, the risk of choosing a poor mapping (compose path) can be reduced by combining several mappings. In our example we merged both the direct mappings and the composed mappings for each pair of data sources. As can be seen from Table 3, in all cases the merged mapping retains the match quality level of the best alternative (either direct matching or using compose).

5.4 Neighborhood Matcher

We evaluate the neighborhood matcher introduced in Section 4 for association mappings of different cardinalities. For the 1:n case we determine a venue same-mapping between DBLP and ACM by utilizing a publication same-mapping. Subsequently we use the determined venue same-mapping for a n:1 neighborhood matching to improve the previous publication same-mapping. Finally we consider the n:m case for author matching using the publication same-mapping.

5.4.1 Venue - Publication (1:n)

In this experiment we use the publication same-mapping between DBLP and ACM determined by the trigram-based attribute match-

Table 4. Matching DBLP-ACM venues using neighborhood matcher based on publication same-mapping (1:n)

Matcher	Attribute (Title)	Neighborhood (Venue)	Merge
	Conference publications		
Precision	96.7%	1.2%	99.2%
Recall	99.8%	100%	98.8%
F-Measure	97.7%	2.4%	99.0%
	Journal publications		
Precision	72.8%	6.5%	99.7%
Recall	95.9%	100%	95.9%
F-Measure	82.8%	12.2%	97.8%
	Overall		
F-Measure	91.9%	3.36%	98.6%

Table 5. Matching DBLP-ACM publications using neighborhood matcher based on venue same-mapping (n:1)

Matcher	Neighborhood (Publication)		
Selection	80%	50%	Best-1
	Conferences		
Precision	100%	100%	94.7%
Recall	100%	100%	100%
F-Measure	100%	100%	97.3%
	Journals		
Precision	100%	99.0%	98.2%
Recall	62.7%	86.4%	100%
F-Measure	77.1%	92.2%	99.1%
	Overall		
F-Measure	80.9%	93.4%	98.8%

er. We also use the venue-publication association mappings for DBLP and ACM within the neighborhood matcher to find a venue same-mapping. Note that the use of attribute matchers based on general string matching is ineffective for finding venue same-mappings. This is because of the high diversity in the value representations of venues, e.g. "VLDB2002" vs. "28th International Conference on Very Large Data Bases"[4].

Table 4 indicates that neighborhood matching very effectively solves this difficult match problem and achieves excellent F-measure values of up to 98.8%, without relying on the combination with other matchers. We also observe that match quality differs between conferences and journals and is also influenced by the selection strategy. For threshold-based selection strategies, conferences can be perfectly matched. This is favored by the fact that the considered conferences have many publications (about 60-120) so that the neighborhood matcher can utilize a large neighborhood for which the majority is correctly matched. The best-1 selection approach does not consider the similarity values and suffered from the missing conference entries for VLDB2002/2003 in ACM DL. We distinguish for the venues between conferences and journals. Conversely, the neighborhood matcher was generally less effective for journals due to their smaller neighborhood, i.e. fewer papers (2-26 per issue). Permissive selection strategies helped to still achieve good match quality in this case since ACM DL covered all relevant journal issues.

5.4.2 Publication - Venue (n:1)

For this experiment we utilize the venue same-mapping determined with the 1:n neighborhood matching and best-1 selection (F-measure 98.8%). The goal is to improve a publication same-mapping between DBLP and ACM. As already discussed in Section 4 the neighborhood matcher determines the corresponding publications of the same venue and is thus alone not adequate for a same-mapping (on average we achieve a recall of 100% and precision of 2%). However, as Table 5 shows the neighborhood matcher can substantially improve the quality of the publication same-mapping. While attribute matching achieves a F-Measure of about 92% the combined use with neighborhood matching raises this value to 98.6%, an excellent result. The improvements are especially large for journals. This is because now the small neighborhoods of journals are positive since they limit the potential match candidates for a publication more than for conferences. In addition, string-based attribute matchers suffer from recurring titles in newsletters like Sigmod Record like editorials, reminiscences on influential papers or interviews. These similarly titled publications appear in different journal issues and can thus be well matched by the neighborhood matcher. The combined use of attribute and neighborhood matcher can very effectively solve almost all such problems.

5.4.3 Author - Publication (n:m)

In this experiment we apply the neighborhood matcher to determine an author same-mapping between DBLP and ACM. The utilized author-publication association is n:m with relatively small but highly variable neighborhoods (about 3 authors per paper on average, variations from 1 author to 27 authors).

Table 5 shows our results for the attribute matcher, neighborhood matcher and their combination by merge (using the Min function). We realize that attribute matching performs already reasonably well. As expected, the neighborhood matcher alone provides rather poor quality. However, as for the n:1 case we can again substantial-

4.Attribute matchers using domain specific knowledge such as conference abbreviations could match venues based on their names.

Table 6. Matching DBLP-ACM authors with the help of neighborhood matcher based on publication same-mapping (n:m)

Matcher	Attribute (Name)	Neighborhood (Publication)	Merge
Precision	99.3%	24.8%	99.9%
Recall	81.3%	99.3%	94.0%
F-Measure	89.4%	39.7%	96.9%

Table 7. Matching DBLP-GS publications with the help of neighborhood matcher based on author same-mapping (n:m)

Matcher	Attribute (Title)	Neighborhood (Author)	Merge
Precision	81.1%	15.2%	85.1%
Recall	81.6%	76.0%	92.9%
F-Measure	81.3%	25.4%	88.9%

Table 8. Matching GS-ACM publications with the help of neighborhood matcher based on author same-mapping (n:m)

Matcher	Attribute (Title)	Neighborhood (Author)	Merge
Precision	86.7%	16.2%	84.6%
Recall	81.7%	75.6%	92.1%
F-Measure	84.1%	26.7%	88.2%

Table 9. Top-5 author duplicate candidates within DBLP

Author	Author	Name	Co-Author	Merge
Catalina Fan	Catalina Wei	64%	100% (4)	82%
Amir M. Zarkesh	Amir Zarkesh	84%	75% (3)	79%
M. Barczyk	M. Barczyc	75%	73% (10)	74%
Agathoniki Trigoni	Niki Trigoni	75%	67% (4)	71%
Joe Chun-Hung Yuen	Joe Yuen	62%	67% (2)	65%

ly increase the overall match quality by combining both attribute and neighborhood matcher to an F-measure of about 97%.

We also applied the n:m neighborhood matcher for improving the publication same-mapping between DBLP and GS based on author-publication associations. For this case, we first had to determine an author same-mapping between GS and DBLP for which we applied an attribute matcher. Note that GS reduces authors' first names to their first letter leading to ambiguous author representations. Another difficulty we observed is that the author lists for GS publications are not always complete, i.e., we have no perfect association mapping. We therefore applied *RelativeLeft* instead of the symmetric *Relative* function for the neighborhood matcher to reduce the influence of missing GS authors. The neighborhood matcher helped to substantially improve the quality of the publications same-mapping from an F-measure of about 81% for attribute matching to almost 89% for the combined use with the neighborhood matching (see Table 7). This is primarily due to the recall increase whereas the precision remains the same. Obviously the author information cannot be used to increase precision, i.e., to distinguish between different publications having the same or a similar title. But on the other hand GS instances with erroneous titles (due to the automatic data extraction process) can still be matched when taking into account the author lists. Similarly, we generated a publication same-mapping between ACM and GS that shows comparative results (Table 8).

5.5 Duplicate Detection

To examine the usefulness of MOMA for the special case of object consolidation within one data source we also experimented with duplicate detection. Although DBLP is curated it contains duplicate authors. As illustrated in Section 4.3 we utilize the co-authorship mapping to identify authors which share a significant number of co-authors.

Table 9 shows the top candidates for being duplicates we found with the co-authorship approach (which is a special case of neighborhood matching). We additionally present the number of compose paths for the neighborhood matcher, i.e., the number of shared co-authors. This value describes the size of the neighborhood and therefore indicates the support by the associated objects. The Trigoni duplicate is already handled by the DBLP website but the underlying publications still contain both notations. The authors Catalina Fan and Catalina Wei are an example for the complexity of identifying author candidates. They share the same list of co-authors and have similar names but we can not determine whether these two authors are the same. Nevertheless, the MOMA framework enabled us to automatically identify such problem cases by combining the co-authorship mapping with author's name string similarity.

Table 10. Summary of matching results (F-Measure)

	Venues	Publications	Authors
DBLP - ACM	98.6%	98.8%	96.9%
DBLP - GS	-	88.9%	-
GS - ACM	-	88.2%	-

5.6 Evaluation Summary

The evaluation has shown that the considered data sources make object consolidation quite challenging due to their different characteristics regarding completeness, dirtiness and accessibility. The MOMA framework proved to be very effective in that we could achieve very high match quality for almost all match tasks, i.e., for matching publications, venues and authors between the considered sources (see Table 10 for a brief summary of the achieved match results). The match strategies proposed in Section 4 were shown to be effective. In particular the combination of several matchers and mappings can compensate weaknesses of individual strategies and are decisive for achieving good quality. The neighborhood matcher proved to be very powerful for association mappings of all cardinalities.

However, the publication match results involving Google Scholar are not yet satisfying. This is influenced by a restrictive evaluation for precision and recall. We required that not only one match per DBLP or ACM publication is found in GS but that all duplicate entries of GS are matched. In future work we will therefore explore match workflows which first determine the duplicates within dirty sources such as Google Scholar and represent them as self-mappings (identifying clusters of duplicate entries). These self-mappings can then be composed with same-mappings between GS and other sources such as DBLP and ACM to find more correspondences.

6. RELATED WORK

Many previous approaches were proposed for object consolidation based on attribute matching. The similarity comparisons between objects typically use generic string distance metrics [2, 10] or domain specific metrics [25]. For example, matching names and addresses is an important aspect for customer relationship management and is therefore supported by many commercial tools [27]. For classifying object pairs as matching or non-matching different techniques have been applied, e.g., rule-based approaches [18] or adaptive learning approaches [34].

Metadata-based mappings have recently found much interest. Especially in P2P systems such mappings and their composition are important for data integration [5]. Systems like Piazza [17] reformulate queries based on metadata mappings. By contrast, our work focuses on instance-level mappings.

In [20] mapping tables are proposed to relate objects from different data sources. By considering mapping tables as constraints new mapping tables can be inferred. Our framework provides semantic mapping types to describe the role of associated objects. Moreover,

MOMA assigns a similarity value to each correspondence and is able to propagate this information by different mapping combiners.

Several authors have recently described approaches for context-based object consolidation. [11] proposes a generic duplicate detection algorithm exploiting information on all associated objects (stored in a dependency graph). Similar approaches can be found in [8, 24, 4]. The key idea is the use of iterative graph algorithms that propagate information about the evidence on matching objects (e.g., represented as edge weights or auxiliary nodes) to related objects. In contrast to this algorithmic work our approach is a framework allowing the specification of match processes and to use of different types of matchers.

In [32] a constraint-based approach for object consolidation is presented. Here, constraints like "If authors X and Y share similar names and some co-authors, they are likely to match" are utilized to match similar objects. Constraints are modeled as conditional probabilities and the proposed algorithms learn a "matching model" to optimize the probabilities. Such constraints refer to associations to other objects in a declarative way, whereas our work leverages this information within a data integration workflow. However, a constraint-based higher level specification could select a cost-based optimized workflow out of the match library.

In [23] the authors utilize amongst others the co-author relationship to identify duplicate authors in DBLP. They apply association rule mining to determine relevant context attributes which are compared afterwards. Similarly, value mappings represent co-occurring attribute values [21]. User feedback and statistical methods like entropy are applied within an iterative procedure. Similar to our work, their approaches generate similarity information from the context of data records. However both are relational approaches in contrast to our mapping based framework.

A special case of associations are hierarchies which are exploited for identifying duplicates [1, 35, 22]. The underlying idea is that matching similarity between two objects also affects the matching similarity of their parents and children. In our work we extended this idea to general association mappings, i.e., hierarchies can be seen as special type of mappings in our framework.

Combining different matchers has shown to be effective for schema matching [13] as well as for object matching. For instance, in [12] so-called profilers incorporating domain or expert knowledge are used to match objects. The combination of their results (e.g., weighted sum of confidence) may overcome the singular weaknesses of each profiler and improve matching accuracy.

Commercial ETL tools [27] support data cleaning workflows but typically do not focus on object matching techniques. Moreover, they typically do not cope with the challenges of matching web documents [33] and perform an offline data cleaning. By contrast, MOMA supports matching of web objects and their dynamic querying from large web sites such as Google Scholar. Furthermore, the mapping-oriented framework supports workflows which reuse existing mappings and utilize associations between objects.

A number of research prototypes of data cleaning / integration frameworks where proposed during the last years, e.g. AJAX [16], TAILOR [15] and Potter's Wheel [28]. Similar to our approach these frameworks allow specifications on how data should be transformed or matched, but they do not make use of instance-based mappings and the reuse of previously determined mappings. Moreover, the frameworks focus on relational databases and offline data cleaning. TAILOR [15] offers a broad variety of tools (e.g., comparison functions or decision models) to build new matchers whereas MOMA provides capabilities for match workflows involving different matchers and reusing match results. Potter's Wheel focuses on interactive data cleaning.

7. SUMMARY & OUTLOOK

We proposed a new framework called MOMA for mapping-based object consolidation. It allows the construction of match workflows combining the results of several matcher algorithms and existing mappings. Different operators for combining fuzzy mappings are supported. Several match strategies including a powerful neighborhood matcher have been proposed to re-use existing mappings of different types and cardinality. A comprehensive evaluation on real data sources demonstrated the flexibility and high effectiveness of the MOMA framework and the proposed match workflows. We also showed that the approaches are applicable for distributed and centralized object matching tasks, e.g. for data integration and duplicate detection.

In future work, we will apply our framework to additional domains such as e-commerce or bioinformatics data sources. We will investigate different match strategies w.r.t. to the underlying applications. For example, small-size online matching strategies are required for ad-hoc data integration applications (possibly accepting lower data quality) whereas large-scale offline matching strategies may utilize multiple matchers to achieve high data quality. In addition we plan to develop approaches for automatically tuning match workflows, in particular to select existing mappings, matchers and combiners and their parameters. For example, machine learning techniques could be utilized to adjust appropriate similarity combination functions for our mapping combination operators automatically.

8. REFERENCES

[1] Ananthakrishna, R., Chaudhuri, S., and Ganti, V.: *Eliminating Fuzzy Duplicates in Data Warehouses*. Proc. of Int. Conf. on Very Large Databases (VLDB), 2002

[2] Bilenko, M., and Mooney, R.: *Adaptive duplicate detection using learnable string similarity measures*. Proc. of Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD), 2003

[3] Bitton, D., and DeWitt, D.J.: *Duplicate Record Elimination in Large Data Files*. In ACM Transactions on Database Systems (TODS) 8(2), 1983

[4] Bhattacharya, I., and Getoor, L.: *Iterative record linkage for cleaning and integration*. Proc. of Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), 2004

[5] Bernstein, P. A., Giunchiglia, A., Kementsietsidis, F., Mylopoulos, J., Serafini, L., and Zaihrayeu, I.: *Data Management*

for Peer-to-Peer Computing : A Vision. Proc. of Int. Workshop on the Web & Databases (WebDB), 2002

[6] Chaudhuri, S., Ganjam, K., Ganti, V., and Motwani, R.: *Robust and Efficient Fuzzy Match for Online Data Cleaning*. Proc. of Int. Conf. on Management of Data (SIGMOD), 2003

[7] Chaudhuri, S., Ganti, V., and Motwani, R.: *Robust Identification of Fuzzy Duplicates*. Proc. of Int. Conf. on Data Engineering (ICDE), 2005

[8] Chen, Z., Kalashnikov, D. V., and Mehrotra, S.: *Exploiting relationships for object consolidation*. Proc. of Int. Workshop on Information Quality in Information Systems (IQIS), 2005

[9] Cohen, W. W., Kautz, H. A., and McAllester, D. A.: *Hardening soft information sources*. Proc. of Int. Conf. on Knowledge Discovery and Data Mining (KDD), 2000

[10] Cohen, W. W., Ravikumar, P., and Fienberg, S. E.: *A Comparison of String Distance Metrics for Name-Matching Tasks*. Proc. of Workshop on Information Integration on the Web (IIWeb), 2003

[11] Dong, X., Halevy, A., and Madhaven, J.: *Reference Reconcilliation in Complex Information Spaces*. Proc. of Int. Conf. on Management of Data (SIGMOD), 2005

[12] Doan, A., Lu, Y., Lee, Y., and Han, J.: *Object Matching for Information Integration: A Profile-Based Approach*. Proc. of Workshop on Information Integration on the Web (IIWeb), 2003

[13] Do, H.-H., and Rahm, E.: *COMA - A System for Flexible Combination of Schema Matching Approaches*. Proc. of Int. Conf. on Very Large Databases (VLDB), 2002

[14] Do, H.-H., and Rahm, E.: *Flexible Integration of Molecular-Biological Annotation Data: The GenMapper Approach*. Proc. of Int. Conf. on Extending Database Technology (EDBT), 2004

[15] Elfeky, M. G., Elmagarmid, A. K., and Verykios, V. S.: *TAILOR: A Record Linkage Tool Box*. Proc. of Int. Conf. on Data Engineering (ICDE), 2002

[16] Galhardas, H., Florescu, D., Shasha, D., and Simon, E.: *AJAX: An Extensible Data Cleaning Tool*. Proc. of Int. Conf. on Management of Data (SIGMOD), 2000

[17] Halevy, A. Y., Ives, Z. G., Mork, P., and Tatarinov, I.: *Piazza: data management infrastructure for semantic web applications*. Proc. of International World Wide Web Conference (WWW), 2003

[18] Hernandez, M. A., and Stolfo, S. J.: *The Merge/Purge Problem for Large Databases*. Proc. of Int. Conf. on Management of Data (SIGMOD), 1995

[19] Jonas, J.: *Identity resolution: 23 years of practical experience and observations at scale*. Proc. of Int. Conf. on Management of Data (SIGMOD), 2006

[20] Kementsietsidis, A., Arenas, M., and Miller, R.J.: *Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues*. Proc. of Int. Conf. on Management of Data (SIGMOD), 2003

[21] Kang, J., Han T.S., Lee, D., and Mitra, P.: *Establishing value mappings using statistical models and user feedback*. In Proc. of Int. Conf. on Information and Knowledge Management (CIKM), 2005

[22] Kailing, K., Kriegel, H.-P., Schönauer, S., and Seidl, T.: *Efficient Similarity Search for Hierarchical Data in Large Databases*. Proc. of Int. Conf. on Extending Database Technology (EDBT), 2004

[23] Lee, M.-L., Hsu, W., and Kothari, V.: *Cleaning the Spurious Links in Data*. In IEEE Intelligent Systems 19(2), 2004.

[24] Parag, and Domingos, P.: *Multi-Relational Record Linkage*. Proc. of Workshop on Multi-Relational Data Mining (MRDM), 2004

[25] Quass, D., and Starkey, P.: *Record Linkage for Genealogical Databases*. Prof. of Workshop on Data Cleaning, Record Linkage, and Object Consolidation, 2003

[26] Rahm, E., and Bernstein, P.A.: *A Survey of Approaches to Automatic Schema Matching*. In VLDB Journal 10(4), 2001

[27] Rahm, E., and Do, H.-H.: *Data Cleaning: Problems and Current Approaches*. In IEEE Data Engineering Bulletin, 23(4), 2000

[28] Raman, V., and Hellerstein, J. Potter's Wheel: *An Interactive Data Cleaning System*. Proc. of Int. Conf. on Very Large Data Bases (VLDB), 2001

[29] Rahm, E., and Thor, A.: *A citation analysis for database publications*. In SIGMOD Record 34(4), 2005

[30] Rahm, E., Thor, A., Aumueller, D., Do, H.-H., Golovin, N., and Kirsten, T.: *iFuice - Information Fusion utilizing Instance Correspondences and Peer Mappings*. Proc. of Int. Workshop on the Web & Databases (WebDB), 2005

[31] Sayyadian, M., Lee, Y., Doan, A., Rosenthal:, A. *Tuning Schema Matching Software using Synthetic Scenarios*. Proc. of Int. Conf. on Very Large Databases (VLDB), 2005

[32] Shen, W., Li, X., and Doan, A.: *Constraint-Based Entity Matching*. Proc. of National Conf. on Artificial Intelligence (AAAI), 2005

[33] Sismanis, Y., Reinwald, B., and Pirahesh, H.: *Document-Centric OLAP in the Schema-Chaos World*. Proc. of Int. Workshop on Business Intelligence for the Real Time Enterprise (BIRTE), 2006

[34] Tejada, S., Knoblock, C.A., and Minton, S.: *Learning domain-independent string transformation weights for high accuracy object identification*. Proc. of Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD), 2002

[35] Weis, M., and Naumann, F.: *DogmatiX tracks down Duplicates in XML*. Proc. of Int. Conf. on Management of Data (SIGMOD), 2005