Appeared in *International ACM SIGMOD Workshop on Information Quality in Information Systems (ACM IQIS)*, June 2005

Exploiting relationships for object consolidation*

Zhaoqi Chen Dmitri V. Kalashnikov Sharad Mehrotra

Computer Science Department
University of California, Irvine

ABSTRACT

Data mining practitioners frequently have to spend significant portion of their project time on data preprocessing before they can apply their algorithms on real-world datasets. Such a preprocessing is required because many real-world datasets are not perfect, but rather they contain missing, erroneous, duplicate data and other data cleaning problems. It is a well established fact that, in general, if such problems with data are not corrected, applying data mining algorithm can lead to wrong results. The latter is known as the "garbage in, garbage out" principle. Given the significance of the problem, numerous data cleaning techniques have been designed in the past to address the aforementioned problems with data.

In this paper, we address one of the data cleaning challenges, called *object consolidation*. This important challenge arises because objects in datasets are frequently represented via descriptions (a set of instantiated attributes), which alone might not always uniquely identify the object. The goal of *object consolidation* is to correctly consolidate (i.e., to group/determine) all the representations of the same object, for each object in the dataset. In contrast to traditional domain-independent data cleaning techniques, our approach analyzes not only object features, but also additional semantic information: *inter-objects relationships*, for the purpose of object consolidation. The approach views datasets as attributed relational graphs (ARGs) of object representations (nodes), connected via relationships (edges). The approach then applies graph partitioning techniques to accurately cluster object representations. Our empirical study over real datasets shows that analyzing relationships significantly improves the quality of the result.

1. INTRODUCTION

Nowadays data mining techniques are widely used to analyze data for scientific applications and business decision making. To build proper models and compute accurate results it is important that analyzed datasets are accurately represented and interpreted. Many real-world datasets however are not perfect, they frequently contain various data cleaning issues such as incomplete, erroneous and duplicate data, which need to be addressed before data mining techniques can be applied. As a result, data mining practitioners frequently spend significant effort on preprocessing of data to address cleaning issues that exist in their datasets, to ensure high quality of the results.

In this paper, we address one common data cleaning challenge known as *object consolidation* [8, 20, 22, 25]. It arises most frequently when the dataset being processed is constructed by merging various data sources into a single unified database, such as by crawling the web. In many real-world datasets objects/entities are not represented by unique identifiers, instead an object is represented by a *description* (a set of instantiated attributes), used in a certain *context*, which may lead to ambiguity. An object might have multiple different representations in the dataset and also an object representation, in general, might match the description of multiple objects instead of one. The goal of object consolidation is to correctly group all the representations that refer to the same object.

For example, consider a database that contains information about two people: 'John A. Smith' and 'John B. Smith'. Firstly, entity 'John A. Smith' might have multiple representations throughout the dataset: e.g., 'John Smith', 'J. Smith' 'John Smithx' (a misspelled representation). Secondly, representation 'J. Smith' can refer to both 'John A.' and 'John B.' Smith, so one representation matches the descriptions of multiple entities. Finally, the fact that there are only two 'John Smith"s in the dataset might not be known in general. Thus, for this example the goal is to determine that all 'John Smith' representations should be clustered into two groups and then assign them to groups such that all representations for 'John A. Smith' are in one group and for 'John B. Smith' in the other. Sometimes it is possible to infer more attributes/information from the *context* in which representations appear. For example, for 'J. Smith' used in a specific context, it might be known that the mentioned 'J. Smith' works at MIT. This context information can be potentially used to consolidate representations better.

Let us use an example to demonstrate the implication of applying data analysis techniques on datasets where the object consolidation problem is not resolved correctly. Consider the task of computing author impact in a citation network using a simple citation-count statistic. That is, the

*RelDC project (`http://www.ics.uci.edu/~dvk/RelDC`)

†This work was supported by NSF grants 0331707, 0331690

IQIS '05, Baltimore, MD, USA

task might be to compute the impact of 'John A. Smith' by counting the number of citations of his publications. This simple task might be more difficult than it seems due to the problem with representations identified above. Notice, even though the representations appear in some context, the information about the object available from the context might be of a limited nature, which makes the object consolidation task challenging. For instance, the only direct information available about the authors, for some of the publications, might be only their first initials and last names. Because of such problems with representations, some of the papers written by 'John A. Smith' might be wrongfully assigned to other authors and some of the papers written by other authors might be assigned to 'John A. Smith'. Thus, the impact of 'John A. Smith', computed on such a dataset, can be very different from the real one.

While the object consolidation problem exists in different domains, in this paper we will often use citation networks, like in the example above, to illustrate our domain-independent approach.

The problem of object consolidation is related to the problem of *record deduplication* or *record linkage* [1, 10, 13, 19, 23] that often arises when multiple tables (from different data sources) are merged to create a single database. The causes of record linkage are similar, i.e. differences in representation of objects across different datasets, entry errors, etc. The difference between the two problems is that while record linkage deals with records in a table, object consolidation deals with entities/objects – a semantic concept of a higher level. In record linkage it is often assumed that many attributes are available in each record, which are very effectively employed for deduplications. In object consolidation, however, very few attributes can be available, thus making the problem more challenging.

Another related problem is the problem of *reference disambiguation* [14, 18]. In the problem of reference disambiguation the goal is to match object representations with the list of possible objects which is known in advance and known to be clean. The requirement of having such a clean list of objects limits the applicability of reference disambiguation. As a rule, each instance of the reference disambiguation problem can be formulated as an instance of the object consolidation problem, while the reverse is not true. That is, the object consolidation problem is more general.

Most of the traditional domain-independent data cleaning techniques belong to the class of *feature-based similarity (FBS)* methods.[1] To determine if two objects/records are the same they employ a similarity function that compares values of object/record attributes (features) for the purpose of deduplication. The values of the attributes of an object are typically derived from the object representation and the context in which it is used. In this paper, we study a domain-independent approach that utilizes not only features but also additional semantic information present in datasets: inter-object (chains of) relationships. For instance, 'J. Smith' might be used to refer to an author in the context of a particular publication. This publication might also have more authors, which can be linked to their affiliated organizations and so on, forming a web of entities inter-connected via relationships. The knowledge of relationships can be exploited alongside attribute-based similarity resulting in improved accuracy of object consolidation. Our approach is based on the following hypothesis, which is referred to as the Context Attraction Principle (CAP):

The CAP hypothesis:

- *if two representations refer to the same entity, there is a high likelihood that they are strongly connected to each other through multiple relationships, implicit in the database;*
- *if two representations refer to different entities, the connection between them via relationships is weak, compared with that of the representations that refer to the same entity.* □

Our approach views the underlying database as an attributed relational graph (ARG), where nodes correspond to object representations and edges correspond to relationships. Our technique first uses feature-based similarity, to determine if two representations can refer to the same objects. If, based on the FBS similarity, two representations can refer to one object, then the relationships between those representations are analyzed to measure the *connection strength* between them. Graph partitioning techniques are then employed to consolidate the representations of objects based on the FBS similarity and connection strength among them.

The **primary contributions** of this paper are:

- A novel object consolidation approach, which, unlike traditional techniques, employs not only attribute (feature) similarity, but also analyzes inter-object relationships to improve the quality of consolidation (Section 4).
- Novel metrics to analyze the quality of the outcome (Section 4.3).
- An empirical evaluation of the proposed technique, that establishes that analyzing relationships is important for object consolidation (Section 5).

Next, in Section 2, we present a motivational example and then, in Section 3, we formalize the problem and introduce the notation necessary to explain the approach.

2. MOTIVATING EXAMPLE

In this section we use an instance of the "author matching" problem to illustrate that exploiting chains of relationships, that exist among entities, can improve the quality of object consolidation.

Consider a toy database consisting of the *author* and *publication* records shown in Figures 1 and 2 on the facing page. Assume that the publications are represented in the database using the attributes ⟨`id,title,authorRef1`, ..., `authorRefN`⟩, where `id` is the paper identifier, `title` is the paper title, and `authorRef`'s are the names of the authors of the paper. Suppose that the author information is stored in the form ⟨`id,authorName,affiliation`⟩, where `id` is the author identifier, `authorName` and `affiliation` are the author's name and affiliation.

[1] For example, two strings 'J. Smith' and 'John Smith', while not identical, are sufficiently similar to suggest that one can be the other and FBS techniques can detect that. It can also be known from the context that the mentioned 'J. Smith' works at MIT and 'John Smith' works at MIT, then FBS approaches can use this additional attribute (affiliation) and suggest that they are now more confident that the two representations refer to the same person.

⟨$P1$, Title1, **'John Smith'**, 'Alan White'⟩
⟨$P2$, Title2, 'Alan White', 'Mike Black'⟩
⟨$P3$, Title3, **'J. Smith'**, 'Mike Black'⟩
⟨$P4$, Title4, 'Tom Grey', **'John Smith'**⟩
⟨$P5$, Title5, 'Tom Grey', 'Kate Red'⟩

Figure 1: *Publication* records

⟨$A1$, **'John Smith'**, 'MIT'⟩
⟨$A2$, **'John Z. Smith'**, 'CMU'⟩
⟨$A3$, **'John Smith'**, 'Stanford'⟩
⟨$A4$, 'Alan White', 'MIT'⟩
⟨$A5$, 'Mike Black', 'NEC'⟩
⟨$A6$, 'Tom Grey', 'Intel'⟩
⟨$A7$, 'Kate Red', 'Stanford'⟩

Figure 2: *Author* records

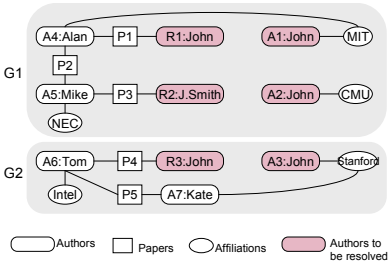


Figure 3: Graph for toy database.

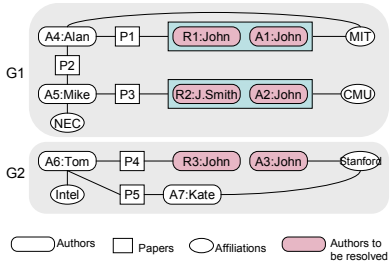


Figure 4: Adding context info.

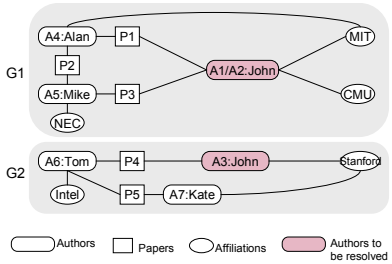


Figure 5: Adding relshp analysis.

We assume that in this database we are only uncertain about representations of *people*. For instance, we are uncertain to which author the representation 'John Smith' in the paper $P1$ refers to: A_1, A_2, or A_3. For that matter, we are also uncertain, whether A_1, A_2, or A_3 are representations of different people, or they are just duplicate records. The latter can be the case, for instance, if 'John Smith' was a graduate student at MIT and then became a faculty at CMU, so $A1$ and $A2$ are duplicates in this scenario. However, under our assumptions, we are certain that the representation 'MIT' in $A1$ is the same organization as the 'MIT' in $A4$, because these two representations are of the type *affiliation*, and not of the type *people*.

Traditional FBS techniques. To solve this problem using traditional techniques one would first try to deduplicate *author* records. After that, those records would be assumed to uniquely represent each distinct author and the goal would be to match `authorRef`'s in *publication* records to the correct authors. For example, existing feature-based similarity techniques can be used to compare the description in each `authorRef` in *publication* records with the values of the `authorName` attribute in *authors* records. Using this technique we can accurately consolidate most of the representations for our toy database, except for the 'John Smith' representations. For example, the author represented as 'Alan White' in publications $P1$ and $P2$ will be mapped uniquely to the author record $A4$ for 'Alan White'; 'Mike Black' in publications $P2$ and $P3$ will be correctly mapped to $A5$ and so on. The only difficulty will be with 'John Smith' and 'J. Smith' in $P1$, $P3$, and $P4$, since each of them can correspond to either $A1$, $A2$, or $A3$.

Let us note that we can visualize the resulting dataset as a graph. In this graph, each entity/object, as well as the not-yet-consolidated representations, become nodes. The relationships that exist among them are visualized as edges.

The graph for the toy database is illustrated in Figure 3. For instance, since author $A1$ is affiliated with 'MIT', there is an edge between them, that corresponds to this relationship. Since, under our assumptions, 'MIT' uniquely identifies the corresponding entity, only one node is created for all 'MIT' representations. However, each 'John Smith' has a separate node in this figure.

Employing context. The most recent data cleaning techniques, such as [7], are also capable of employing the *context* to improve the quality of cleaning. There might be additional context information ('context attributes') that such algorithms might be able to use.

For instance, if it is also known that the coauthor of $P1$, 'Alan White', is from 'MIT', then, given that $A1$ is from 'MIT' as well, we may decide that 'John Smith' in $P1$ refers to $A1$ and not to $A2$ or $A3$. As another example, if we already know that $A2$ has papers with titles similar to that of $P3$, then we can infer that 'J. Smith' in $P3$ refers to $A2$. The resulting dataset can be visualized as the graph illustrated in Figure 4, where $R1, A1$, and $R2, A2$ are shown to be merged into two nodes.

Analyzing relationships. Now we will show how additional semantic information, stored in the relationships that exist between entities, can help to improve the quality of cleaning even further.

Observation 1: ('John Smith' in $P1$). First, to handle author 'John Smith' in $P1$, we observe that his co-author 'Alan White' has also written a paper $P2$ with 'Mike Black', who in turn has a paper $P3$ with 'J. Smith'. This gives us certain evidence that 'John Smith' in $P1$ is the same person as 'J. Smith' in $P3$. The intuition behind it is that people in the similar/related research areas tend to cooperate with each other and form co-authorship networks. Based on this evidence, we might decide that $P1$ and $P3$ are written by the same author, whose name is 'John Smith'.

Recall, by using the context, we have determined above that 'John Smith' in $P1$ refers to $A1$, and 'J. Smith' in $P3$ to $A2$. Therefore, the evidence suggests that $A1$ and $A2$ are duplicate records for the same author – the fact that was not captured by the above feature-based similarity algorithm!

Observation 2: ('John Smith' in $P4$). Consider the task of deciding whether the representation 'John Smith' in $P4$ refers to $A1$, $A2$, or $A3$. Observe that the coauthor of that paper, 'Tom Grey', has a paper $P5$ with 'Kate Red', who is at 'Stanford'. The author $A3$ is also at 'Stanford', and thus we are able to establish a connection between 'John Smith' in $P4$ and $A3$. Given that there are no such connections to $A1$ an $A2$ that we can find, we might decide that 'John Smith' in $P4$ probably refers to $A3$.

Generic approach. At first glance, the analysis in Observations 1 and 2 might seem to be domain-specific. How-

ever, a domain-independent approach emerges if we view the underlying database as a graph of object representations (modeled as nodes) linked to each other via relationships (modeled as edges). The analysis can be viewed as application of the CAP hypothesis to the toy database, represented as the graph in Figure 4.

The first observation we made, regarding $R1$ and $R2$ being two representations for the same author, was based on the presence of the path (we use 'path', 'relationship chain', and 'connection' interchangeably):

$$R1 \rightarrow P1 \rightarrow A4 \rightarrow P2 \rightarrow A5 \rightarrow P3 \rightarrow R2.$$

Via this path, we were able to 'connect' $R1$ and $R2$.

The second observation we made regarding disambiguation of 'John Smith' in $P4$ was based on the presence of the path

$$R3 \rightarrow P4 \rightarrow A6 \rightarrow P5 \rightarrow A7 \rightarrow \text{'Stanford'} \rightarrow A3.$$

Via this path, we 'connected' $R3$ and $A3$.

Figure 5 shows the resulting graph, which reflects the outcome of the above analysis. In this figure, $R1$, $R2$, $A1$, and $A2$ are grouped together (forming the first group), since, those representations are likely to refer to the same person. Similarly, $R3$ and $A3$ are grouped as well, forming the second group. Let us observe that the graph in Figure 4 can be split into two connected subgraphs $G1$ and $G2$, such that $G1$ contains nodes of the first group and $G2$ of the second group.

In general, connections between representations can be more complex than those in this example. Therefore, a similar analysis may need to measure and compare the "strength" in the connections that exists between various entities.

Thus, the generic approach for object consolidation may consist of the following steps. First, the approach identifies the representations that can refer to the same entity. Then, it discovers *connections* between these representations and measures the connection strength between them to obtain the evidence to be used in the consolidation process. The algorithm then employs a graph partitioning algorithms to group the representations into clusters, such that the connections among the nodes in the same cluster are strong, and among the nodes across the clusters are weak, to satisfy the CAP principle.

Naturally, one should demonstrate that the CAP hypothesis holds over real datasets by designing a generic solution to exploiting relationships for object consolidation. We will develop one such general domain-independent strategy in Section 4. We perform an extensive study of the proposed approach over a real dataset, to establish that exploiting relationships can further improve the quality of object consolidation. Before we develop our solution, we first introduce the notation and concepts needed to explain the approach, in Section 3.

3. PROBLEM FORMULATION

3.1 Notation

Let $\mathcal{D}$ be the database being processed. We will use $O = \{o_1, o_2, \ldots, o_{|O|}\}$ to denote the set of all entities (or, *objects*) in $\mathcal{D}$. 'Entities' here have the same meaning as in the E/R model. Various consolidation scenarios are possible w.r.t. O. In one scenario, the consolidation algorithm has some information about the objects in O and about the cardinality of O. A more complicated scenario is when no information about the number or the nature of the objects in O is available.

Entities are referred in the database via *representations*. Let $X = \{x_1, x_2, \ldots, x_{|X|}\}$ (where $|X| \geq |O|$) be the set of all representations in $\mathcal{D}$. A representation is a description of an entity, which may consist of one or more attributes. For instance, in the toy database in Section 2, `authorRef` representations consist of only one attribute $\langle$*author name*$\rangle$. If, besides author names, author affiliation were also stored in the *publication* records, then `authorRef` references would have consisted of two attributes – $\langle$*author name*, *author affiliation*$\rangle$.

Each representation x_i semantically refers to a single specific entity in O, which we denote by $d[x_i]$, where $d[x_i]$, in general, is unknown to the consolidation algorithm. The **goal** is to group all the representations in X into a set C of $|O|$ non-empty clusters, $C = \{C_1, C_2, \ldots, C_{|O|}\}$, such that all the representations in one cluster refer to the same entity, and no two representations from two different clusters refer to the same entity.[2] That is, for any two representations x and y from the cluster C_i it should follow that $d[x] = d[y]$. Similarly, for any two representations x and y from two distinct clusters C_i and C_j, it should follow that $d[x] \neq d[y]$.

We will use $C[x_i]$ to denote the *group set* of x_i – the set of all the representations from X that refer to the same entity as x_i, and thus should be put into the same group with x_i: $C[x_i] = \{x_j \in X : d[x_j] = d[x_i]\}$. Similar to $d[x_i]$, the group set $C[x_i]$, in general, is unknown to the consolidation algorithm. Given this notation, the goal can be reformulated as determining $C[x_i]$ for each $x_i \in X$. Let $S[x_i]$ denote the *consolidation set* of x_i – the set of all representations from X such that x_i and any representation from $S[x_i]$ can potentially refer to the same entity based on their feature-based similarity. That is, $S[x_i] = \{x_j : sim(x_j, x_i) > \tau\}$, where sim denote a feature-based similarity function and τ is some threshold. We assume that $C[x_i] \subseteq S[x_i]$.

To illustrate these concepts, consider the database in Table 1 on the next page. It contains four representations of people: x, x_A, x_B, and y. Assume that those representation are not misspelled, and therefore x_A and x_B cannot refer to the same person, i.e. $d[x_A] \neq d[x_B]$. Suppose that x and x_A are representations of one person, x_B is of another person, and y is of a third person. Then the corresponding cluster sets and consolidation sets are shown in Table 1.

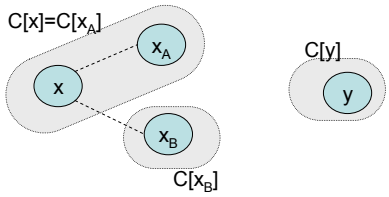


Figure 6: Similarity edges.

Figure 6 graphically illustrate this example. In this figure, a node is created per representation and a *similarity* edge is created between two nodes only if the corresponding

[2]Notice, the goal is only to be able to accurately group representations, **not** to infer any information about the entities they represent, etc.

Representations	C	S	S^*
$x =$ 'J. Smith'	$C[x] = \{x, x_A\}$	$S[x] = \{x, x_A, x_B\}$	$S^*[x] = \{x, x_A, x_B\}$
$x_A =$ 'J. A. Smith'	$C[x_A] = \{x, x_A\}$	$S[x_A] = \{x, x_A\}$	$S^*[x_A] = \{x, x_A, x_B\}$
$x_B =$ 'J. B. Smith'	$C[x_B] = \{x_B\}$	$S[x_B] = \{x, x_B\}$	$S^*[x_B] = \{x, x_A, x_B\}$
$y =$ 'Alan White'	$C[y] = \{y\}$	S[y] = {y}	$S^*[y] = \{y\}$

Table 1: Example to illustrate the notation.

representations may refer to the same entity, as determined by their feature-based similarity.

In general, such a *virtual similarity graph* can be created for the set X of all the representations in $\mathcal{D}$. Typically, this graph is composed of multiple *virtual connected subgraphs (VCS)*. To define the VCS for a representation x_i, we first define the set of all the representations $S^*[x_i]$ that belong to the same VCS as x_i. The set $S^*[x_i]$, consists of x_i and each representation $y \in X$, such that there exist a path $x_i \rightsquigarrow y$, consisting of only similarity edges. Notice that $C[x_i] \subseteq S[x_i] \subseteq S^*[x_i]$, as illustrated in Table 1. For instance, the graph in Figure 6 consists of two VCS's: one with nodes $\{x, x_A, x_B\}$ and the other one with node $\{y\}$.

The concept of a VCS is useful because the representations that belong to different VCS's cannot refer to the same entity. This allow us to cluster references in a "one VCS at a time" fashion.

3.2 The Attributed Relational Graph

Our consolidation approach views the database $\mathcal{D}$ as an undirected attributed relational graph (ARG) $G = (V, E)$, where V is the set of nodes and E is the set of edges. ARGs are often utilized by various applications to represent the entities (nodes) in a dataset, connected via relationships (edges). The graph is called *attributed* because attributes can be associated with both the edges and nodes of such a graph. We use ARGs in a similar manner, as elaborated below.

Nodes. In real-world datasets all the representations can be divided into two categories: those for which consolidation is trivial and those for which this task requires extra processing. For instance, in the toy database all representations of *affiliations* can be trivially consolidated by applying feature-based similarity since this similarity was sufficient to uniquely identify each distinct entity. However, consolidation of certain *author* representations required an additional processing. For those cases where consolidation is trivial, the representations of the same entity are clustered, and a node is created per the resulting cluster of representations. For those cases where consolidation is not trivial, a node is created per representation. Figure 4 illustrates the resulting graph for the toy database from Section 2. Let us note that the way our approach creates nodes closely resembles the way they are typically created in ARGs – to represent distinct entities.

Edges. The approach handles two types of edges: *regular* and *similarity* edges. Regular edges connect representations of entities if they are related via relationships.[3] For instance, in the graph in Figure 3 in Section 2, edges connect all the representations of authors and the papers they have written. A *similarity* edge is created for each pair of representation that can refer to the same entity (based on feature-based similarity). Each similarity edge has a weight associated with it (a real number from [0,1] interval) which reflects the degree of similarity between the two representations based on their features. Similarity edges for the toy database are illustrated in Figure 7.

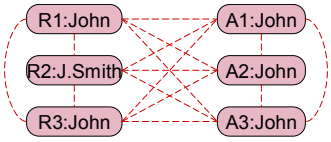


Figure 7: Virtual connected subgraph

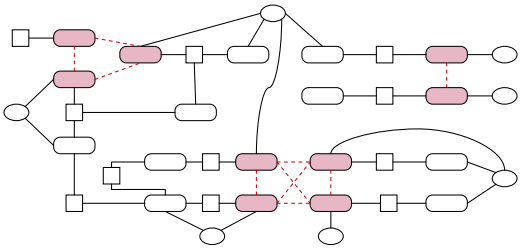


Figure 8: Graph Example

Representing nodes and edges graphically. We will use solid lines to graphically represent regular edges and broken lines for similarity edges. Nodes that correspond to already consolidated clusters of representations will not have color; representations that are yet to be consolidated are represented as shaded nodes. So an ARG might look like the one illustrated in Figure 8. Notice how the similarity edges define the three distinct VCS's in this example.

3.3 Connection Strength

In Section 1, we discussed that the approach consolidates representations based on the CAP hypothesis. To achieve that, it utilizes the notion of *connection strength* between two representations x and y, denoted as $c(x, y)$. This measure captures how strongly x and y are connected to each other via relationships. Many different models for computing $c(u, v)$ have been proposed in the literature, and we will take up one of them in Section 4.1.

4. THE CONSOLIDATION ALGORITHM

We now have developed all the concepts and notation needed to explain our approach for object consolidation. The approach exploits both features and relationships for the purpose of consolidation, and outputs the resulting clus-

[3]We will concentrate primarily on binary relationships. Multiway relationships are rare and most of them can be converted to binary relationships [11]. Most of the design models/tools only deal with binary relationships, for instance ODL (Object Definition Language) supports only binary relationships.

tering as the outcome. The approach consolidates representations using the following steps:

1. **Construct the ARG and identify all VCS's.** The first step is to construct the ARG for the dataset. We assume that feature-based similarity is used in constructing such a graph. This step has been explained in detail in Section 3.2.

2. **Choose a VCS and compute $c(u, v)$'s.** Pick a VCS in the ARG to be partitioned next. Then, compute the connection strength $c(u, v)$ for each pair of representations u and v in the VCS that are connected via a similarity edge.

3. **Partition the VCS.** Take, from Step 2, the VCS and the connection strength values. Use a graph partitioning algorithm to partition the VCS into clusters. The partitioning is carried out based on the connection strengths. After the VCS is partitioned, adjust the ARG accordingly. If the VCS was the last to be partitioned, then stop. Otherwise, go to Step 2.

We now discuss the above steps in more detail in the following subsections.

4.1 Computing Connection Strength

The connection strength measure $c(u, v)$ for two objects u and v computes how strongly they are connected to each other via relationships.

4.1.1 Existing models.

Recently, there has been a spike of interest by various research communities in the measures directly related to the $c(u, v)$ measure. Since the $c(u, v)$ measure is at the core of the proposed consolidation approach, we next analyze several principal existing models for computing $c(u, v)$.

Diffusion Kernels. The earliest work in this direction that we can trace is in the area of kernel-based pattern analysis [29]. The kernel methodology currently undergoes very active development, and shows a great promise for improving various pattern analysis tasks. In particular, this area studies 'diffusion kernels on graph nodes', which are of direct interest in our context and are defined as follows.

A *base similarity graph* $G = (S, E)$ for a dataset S is considered. The vertices in the graph are the data items in S. The undirected edges in this graph are labeled with a 'base' similarity $\tau(\mathbf{x}, \mathbf{y})$ measure. That measure is also denoted as $\tau_1(\mathbf{x}, \mathbf{y})$, because only the direct links (of size 1) between nodes are utilized to derive this similarity. The base similarity matrix $\mathbf{B} = \mathbf{B}_1$ is then defined as the matrix whose elements $\mathbf{B}_{\mathbf{xy}}$, indexed by data items, are computed as $\mathbf{B}_{\mathbf{xy}} = \tau(\mathbf{x}, \mathbf{y}) = \tau_1(\mathbf{x}, \mathbf{y})$. Next the concept of base similarity is naturally extended to path of arbitrary length k. To define $\tau_k(\mathbf{x}, \mathbf{y})$, the set of all paths $P^k_{\mathbf{xy}}$ of length k between the data items $\mathbf{x}$ and $\mathbf{y}$ is considered. The similarity is defined as the sum over all these paths of the products of the base similarities of their edges:

$$\tau_k(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}_1 \mathbf{x}_2 \ldots \mathbf{x}_k) \in P^k_{\mathbf{xy}}} \prod_{i=1}^{k} \tau_1(\mathbf{x}_{i-1}, \mathbf{x}_i)$$

Given such $\tau_k(\mathbf{x}, \mathbf{y})$ measure, the corresponding similarity matrix $\mathbf{B}_k$ is defined. It can be shown that $\mathbf{B}_k = \mathbf{B}^k$. The idea behind this process is to enhance the base similarity by those indirect similarities. For example, the base similarity $\mathbf{B}_1$ can be enhanced with similarity $\mathbf{B}_2$, e.g. by considering a combination of the two matrices: $\mathbf{B}_1 + \mathbf{B}_2$. The idea generalizes to more then two matrices. For instance, by observing that in practice the relevance of longer paths should decay, it was proposed to introduce a decay factor λ and define what is known as the *exponential diffusion kernel*: $\mathbf{K} = \sum_{k=0}^{\infty} \frac{1}{k!} \lambda^k \mathbf{B}^k = \exp(\lambda \mathbf{B})$. The *von Neumann diffusion kernel* is defined similarly: $\mathbf{K} = \sum_{k=0}^{\infty} \lambda^k \mathbf{B}^k = (\mathbf{I} - \lambda \mathbf{B})^{-1}$. The diffusion kernels can be computed efficiently by performing eigen-decomposition of $\mathbf{B}$, that is $\mathbf{B} = \mathbf{V}' \mathbf{\Lambda} \mathbf{V}$, where the diagonal matrix $\mathbf{\Lambda}$ contains the eigenvalues of $\mathbf{B}$, and by making an observation that for any polynomial $p(x)$, the following holds $p(\mathbf{V}' \mathbf{\Lambda} \mathbf{V}) = \mathbf{V}' p(\mathbf{\Lambda}) \mathbf{V}$. The elements of the matrix $\mathbf{K}$ exactly define what we refer to as the connection strength: $c(\mathbf{x}, \mathbf{y}) = \mathbf{K}_{\mathbf{xy}}$.

The solutions proposed for the diffusion kernels work well, if the goal is to compute $c(u, v)$ for all the elements in the dataset. They are also very useful for illustration purposes and similar in nature (though cannot be used "as is") to the weight-based model we employ in our previous work [14, 17]. However, in data cleaning, the task is frequently to compute only some of $c(u, v)$'s, thus more efficient solutions are possible. Also, often after computing one $c(u, v)$, the graph is adjusted in some way, which affects the values of $c(u, v)$'s computed after that.

Relevant importance in graphs. White et al. in [33] consider the problem of computing 'relevant importance' of a set of nodes in a graph with respect to the set of 'root' nodes. The problem of computing $c(u, v)$ can be postulated as computing the relevant importance of node u with respect to the root node v. In [33] several known techniques are evaluated for their goal. Basic techniques, such as the length of the shortest paths between u and v, are compared against more involved ones, which are similar to the diffusion kernels. In the context of our problem, the work in [33] has an advantage over the work on kernels, because [33] focuses on efficient computation of one $c(u, v)$, whereas the kernel methods compute the whole similarity matrix $\mathbf{K}$.

Electric circuit analogy. Faloutsos et al. in [9] considers a model for computing $c(u, v)$. They view the graph as an electric circuit consisting of resistors, and compute $c(u, v)$ as the amount of electric current that goes from u to v. One of the primary contributions of that paper is the optimizations that scale their approach to large graphs.

Random walks in graphs. Another common model used for computing $c(u, v)$ is to compute it as the probability to reach node v from node u via random walks in the graph. That model has been studied extensively, including in our previous work [14, 17].

Problems with existing models. In the context of data cleaning, the existing techniques have several disadvantages. One disadvantage is that the true 'base' similarity is rarely known in real-world datasets. Some existing techniques try to mitigate that by *imposing* a similarity model. However, the CAP principle implies *its own* similarity measure, and any imposed model, created for its own sake in isolation from the specific application, might have little to do with it. Ideally, the similarity measure should be derived directly from data for the specific application at hand that employs it. One step toward achieving this, is to consider *parameterized* models and then try to learn an optimal

combination of parameters directly from data. We have explored such an approach in [16] for the problem of reference disambiguation. Next we will present the $c(u, v)$ model we use in this paper for object consolidation. That model is parameterized, however in this paper we do not study how to learn the right parameters, but rather assume they are assigned by the domain analyst, as we will elaborate shortly. Let us note that our overall approach is independent from a particular $c(u, v)$ model, and a different model, e.g. [17], can be utilized for this purpose.

4.1.2 The connection strength model.

The $c(u, v)$ model we use is very similar to that of the diffusion kernels. To compute $c(u, v)$, the set of all L-short simple paths $\mathcal{P}_L(u, v)$ between u and v is analyzed, where a path is L-short if its length does not exceed L. The total connection strength between nodes u and v is computed as the sum of connection strengths of paths in $\mathcal{P}_L(u, v)$:

$$c(u, v) = \sum_{p \in \mathcal{P}_L(u,v)} c(p). \quad (1)$$

The connection strength $c(p)$ of each individual path p is computed the same way as in the kernel formulae: as a product of base similarities of edges.

The differences between the proposed model $\mathcal{M}$ and that of the kernels $\mathcal{M}_K$ are as follows. The model $\mathcal{M}_K$ employs the decay factor λ, whereas $\mathcal{M}$ does not. In $\mathcal{M}_K$, we have $L = \infty$, in $\mathcal{M}$ the parameter L is finite, e.g. $L = 5$. In $\mathcal{M}_K$ all base similarities $\tau(u, v)$ are known, in $\mathcal{M}$ we will derive them in a particular way: the ultimate goal (not considered in this paper) is to eventually to be able to learn them directly from data.

The procedure for computation $c(u, v)$ consists of two logical phases. The first phase discovers connections/paths between u and v. The second phase computes/measures the strength in connections discovered by the first phase.

The connection discovery phase. In general there can be many connections between nodes u and v. Intuitively, many of those (e.g., very long ones) are not very important. To capture most important connections while still being efficient, the algorithm computes the set of all L-short simple paths $\mathcal{P}_L(u, v)$ between nodes u and v in graph G. This algorithm is the bottleneck of the overall approach. Several optimizations of this algorithms has been studied in [14,17], which achieve orders of magnitude in improvement. In this paper we employ the same optimizations.

The second phase computes the strength in the discovered connections using Equation (1). We are yet to specify how we compute the connection strength $c(p)$ of each individual path p from $\mathcal{P}_L(u, v)$ in Equation (1). Let us address this issue.

Motivating $c(p)$ formula. Which factors should be taken into account when computing the connection strength $c(p)$ of each individual path p? Figure 9 illustrates two different paths (or connections) between nodes u and v: $p_a = u \to a \to v$ and $p_b = u \to b \to v$. Let us understand which connection is better.

Both connections have the same length of two. One connection is going via node a and the other via node b. The intent of Figure 9 is to show that node b "connects" many nodes, not just u and v, whereas node a "connects" only u and v. For instance, in the context of the author matching problem u and v can be two authors, a can be a publication and b a university, as illustrated in Figure 10. We argue the connection between u and v via b is much weaker than the connection between u and v via a: since b connects many nodes, it is not surprising we can connect u and v via b as well. Notice, measures such as path length, network flow do not capture the fact that $c(p_a) > c(p_b)$.

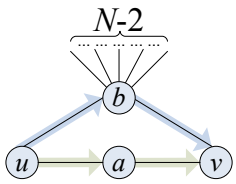


Figure 9: Motivating $c(p)$.

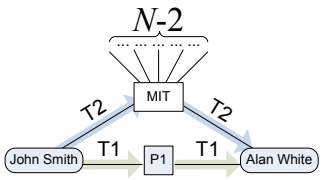


Figure 10: $c(p)$.

Figure 11: Experiments

We compute $c(p)$ as follows. Let us assume first that path p consists of only regular edges, and no similarity edges. All edges in the ARG can be classified into a finite set of types $T = \{T_1, T_2, \ldots, T_m\}$. For example, for the author matching problem, type T_1 can be the edges that connect authors and publications, type T_2 can be the edges that connect authors and their affiliated organizations. So path p_a in Figure 10 can be viewed as a $\langle T_1, T_1 \rangle$ path and path p_b as a $\langle T_2, T_2 \rangle$ path. Each edge type T_i has a weight w_i (a real number from [0,1] interval) associated with it. The connection strength of path p is computed as the product of weights associated with its edge types. For example, if path p contains n_1 edges of type T_1, n_2 edges of type T_2 and so on, then $c(p)$ is computed as

$$c(p) = w_1^{n_1} w_2^{n_2} \times \cdots \times w_m^{n_m}. \quad (2)$$

Assigning weights. An important question is how those w_i weights are determined. There are several methods to accomplish that, we will briefly discuss only two methods:

1. The weights are assigned by the domain analyst.
2. The weights are learned from data using a supervised learning algorithm.

The first case is straightforward: the domain analyst, who is well-familiar with the dataset and the nature of this algorithm, picks the appropriate weights. However, it is often desirable to minimize the participation of the analyst. Thus, in our ongoing work, e.g. [16], we address the second case, where the challenge is to learn the weights automatically, directly from training data, by employing a supervised learning algorithm. Ideally, that solution should lead to a self-tunable algorithm, which achieves the best quality of consolidation.

Example. Without loss of generality, let us assume the weights are assigned by a domain analyst. Then, in the context of our motivating example, taking into account the fact that connections via publications are more unique than those via universities, the analyst might decide that the following combination of weights is reasonable: $w_1 = \frac{1}{2}$ and $w_2 = \frac{1}{10}$. So $c(p_a) = w_1^2 = \frac{1}{4}$, $c(p_b) = w_2^2 = \frac{1}{100}$, and $c(p_a) > c(p_b)$.

Paths with similarity edges. Let us consider paths that can contain similarity edges. Recall that a similarity edge between nodes for two representation x and y denote the fact that there is a chance that x and y can refer to the same

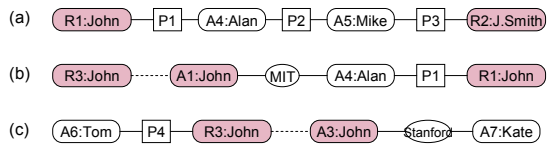


Figure 12: Paths with/without similarity edges.

entity. This edge has a weight w_F associated with it, which reflects the degree of similarity between x and y based on their features: $w_F = sim(x, y)$. We will refer to this weight as the FBS weight.

Let us note that a path containing a similarity edge might not even exist in reality. For instance, in Figure 12(a) the path consists of only regular edges and we are confident it exists. However, the existence of the path in Figures 12(b) depends on whether $R3$ and $A1$ refer to the same entity or not. The same applies to $R3$ and $A3$ in Figure 12(c).

The simplest solution is not to consider paths that contain similarity edges at all. Another (heuristic) solution we use is to associate the same very small *base* weight w_ε for all similarity edges. Then, compute the total weight v_i of a similarity edge as a product of its FBS weight w_F and the base weight w_ε: $v_i = w_F \times w_\varepsilon$. So, if path p has n_i edges of type T_i $(i = 1, 2, \ldots, m)$ and k similarity edges with total weights $v_1, v_2, \ldots, v_k$, then $c(p)$ is computed as

$$c(p) = w_1^{n_1} w_2^{n_2} \times \cdots \times w_m^{n_m} v_1 v_2 \times \cdots \times v_k. \qquad (3)$$

4.2 Consolidating objects by partitioning VCS's

To consolidate objects we need to partition the representations in each VCS. Each VCS contains representations of at least one object. If a representation of an object is contained in a VCS, then, by construction of VCSs, the rest of the representations of the same objects are contained in the same VCS. Two scenarios are possible. In one scenario, the knowledge of the number of objects contained in each VCS is available. In the other scenario, no such knowledge is available.

Assume we know that a given VCS contains the representations of exactly k objects. Then, the VCS must be partitioned into exactly k clusters. We need to consider all possible partitions of the VCS into k clusters, which are feasible according to the similarity edges in the VCS. There might be several such partitions and we should choose the one that best satisfies the CAP principle. We try to achieve that by employing a min-cut algorithm proposed in [30]. The standard min-cut problem is defined as follows. Given a weighted graph $G = (V, E)$, the goal is to partition V into two non-empty disjoint subsets V_1 and V_2, such that the total weight of the edges connecting the two parts, called the *cut* (i.e., the weight of $\{(u, v) \in E : u \in V_1; v \in V_2\}$), is minimized.

In our case, we partition V into not 2, but k subsets. The min-cut algorithm we employ [30] has two useful properties: (a) it utilizes a 'normalized' (to $[0, 2]$ interval) cut; and (b) it achieves the goal of minimizing connections across clusters and maximizing connections inside clusters at the same time. The former property will later allow us to specify a single threshold per all VCS's, while the latter property is important to better satisfy the CAP principle.

Defining weights for partitioning. A min-cut algorithm uses weights $w(u, v)$ between nodes for partitioning. The weights are defined as follows: if there is a similarity edge between u and v then $w(u, v) = c(u, v)$, otherwise $w(u, v) = 0$. Let us note that the formula for $w(u, v)$ can also include the feature-based similarity $sim(u, v)$, e.g. as a weighted sum: $w(u, v) = \alpha \times c(u, v) + (1 - \alpha) \times sim(u, v)$, $0 \leq \alpha \leq 1$. But we do not study this approach in this paper.

Further partitioning. Consider now the second scenario, where we do not know the number of objects in a given VCS. The algorithms handle this case by first partitioning VCS into two parts. Then, the algorithm decides whether to actually split the nodes into two clusters, or not. It achieves that by comparing the value of the resulting normalized cut c against predefined threshold τ. Currently, the value of τ is not yet learned from data, but rather is set by the domain analyst. If $c > \tau$, the two parts are still well inter-connected and the algorithm does not divide VCS further into two clusters. That means the algorithm assumes that all the representations in this VCS refer to a single real-world entity, and hence they should be grouped together in one cluster. On the other hand, if $c < \tau$, the algorithm repeatedly partition the resulting subgraphs until $c > \tau$. The resulting clustering of representations is returned as the final result of the algorithm.

4.3 Measuring the quality of outcome

The goal of object consolidation algorithms is to accurately group the representations of entities. However, consolidation algorithms can make mistakes, and thus the quality of the outcome should be quantified. Measures known as *dispersion* and *diversity* has been proposed before for this purpose [2].

Dispersion. We want the representations of the same entity to be clustered together in one cluster. The *dispersion* of a given entity captures the number of distinct clusters into which its representations are clustered. Therefore, the lesser the dispersion the better and the ideal dispersion is 1 for a given entity.

Diversity. We also want each cluster to contain representations of just one entity. The *diversity* of a given cluster captures the number of distinct entities whose representations are in this cluster. Similarly, the lesser the diversity the better and the ideal diversity is 1 for a given cluster.

Problems with dispersion and diversity. The *dispersion* and *diversity* do not always accurately reflect the quality of the outcome of object consolidation, although they are simple and easy to understand. Consider the following example. Assume a VCS to be partitioned is composed of $2n$ representations $a_1, a_2, \cdots, a_{2n}$ of entity E_A and of $2n$ representations $b_1, b_2, \cdots, b_{2n}$ of entity E_B. The goal is to group them correctly and therefore the ideal result is the two clusters:

$$C_1 = \{a_1, a_2, \ldots, a_{2n}\},$$
$$C_2 = \{b_1, b_2, \ldots, b_{2n}\}.$$

The algorithm however can make mistakes and the resulting two clusters might be:

$$C_1 = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n\},$$
$$C_2 = \{a_{n+1}, a_{n+2}, \ldots, a_{2n}, b_{n+1}, b_{n+2}, \ldots, b_{2n}\},$$

that is, half of the representations are misassigned. In another situation, the two clusters might be

$$C_1 = \{a_1, a_2, \ldots, a_{2n-1}, b_{2n}\},$$

$$C_2 = \{b_1, b_2, \ldots, b_{2n-1}, a_{2n}\},$$

that is, just one is misassigned for each cluster.

Obviously, the latter answer is better than the previous one. However, in both situations, since the representations of both E_A and E_B are scattered across two clusters, the *dispersion* of each of the two entities, E_A and E_B, is 2. The *diversity* of each of the two clusters C_1 and C_2 is 2, since each cluster contains the representations of both E_A and E_B. Therefore, the *dispersion* and *diversity* measures cannot distinguish the two situations in this case.

Entropy-based quality measures. We argue that the metrics known as *entropy* can be utilized to better capture the above situations. Entropy was first proposed by Shannon in his famous work [28] on the mathematical theory of communication.

The *entropy* of a discrete random variable X reflects the degree of uncertainty associated with possible values of X. Let variable X take values from set $\{x_1, x_2, \ldots, x_n\}$ with the respective probabilities $p(x_1), p(x_2), \ldots, p(x_n)$, where $p(x_i) \neq 0$ and $p(x_1) + p(x_2) + \cdots + p(x_n) = 1$. Then the entropy of X, denoted $H(X)$, is defined as:

$$H(X) = \sum_{i=1}^{n} p(x_i)\, log_2 \frac{1}{p(x_i)}$$

Entropy $H(X)$ attains its minimum value $H(X) = 0$, when there exists some i such that $p(x_i) = 1$. In that situation there is no uncertainty associated with X: the value of X is always x_i. On the other hand, $H(X)$ attains its maximum value in the most uncertain scenario: when all the values $x_1, x_2, \ldots, x_n$ are equally likely, in which case $H(X) = log_2\, n$. Thus, $H(X) \in [0, log_2\, n]$.

Entity entropy. Assume that a certain entity E has m representations in the database, which are assigned to n clusters $C_1, C_2, \ldots, C_n$ by the algorithm. The assignment is such that m_1 representations are assigned to the cluster C_1, m_2 to C_2 and so on, such that $m_i \neq 0$ and $m_1 + m_2 + \cdots + m_n = m$. To measure the spread of representations of E over the clusters, we consider the fractions of all representations of E assigned to each cluster C_i $(i = 1, 2, \ldots, n)$: $p_i = \frac{m_i}{m}$. Then, using those fractions, we utilize entropy to quantify the spread of the entity's representations: $H(E) = \sum_{i=1}^{n} p_i\, log_2 \frac{1}{p_i}$. Let us note that the dispersion for this case is always n. The lower the value of $H(E)$, the better. The ideal value is 0.

Cluster entropy. Similarly, we can define the cluster entropy. Assume that the algorithm assigns to a cluster C exactly m representations, that correspond to n entities. Suppose that m_1 of them are representations of entity E_1, m_2 of E_2, ..., m_n of E_n, where $m_i \neq 0$ and $m_1 + m_2 + \cdots + m_n = m$. Like in the case above, we consider fractions $p_i = \frac{m_i}{m}$ and then use them in entropy to quantify the spread of the cluster's representations: $H(C) = \sum_{i=1}^{n} p_i\, log_2 \frac{1}{p_i}$. Let us observe that the diversity in this case is always n. The lower the value of $H(C)$ the better, the ideal value is 0.

Example. Consider the example above with the two clusters C_1 and C_2, and assume that $n = 10$. In the first situation, the entity entropy is:

$$H(E_A) = \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} + \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} = 1,$$
$$H(E_B) = 1,$$
$$avg = \tfrac{H(E_A)+H(E_B)}{2} = 1.$$

The cluster entropy is:

$$H(C_1) = \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} + \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} = 1,$$
$$H(C_2) = 1,$$
$$avg = \tfrac{H(C_1)+H(C_2)}{2} = 1.$$

In the second situation, the entity entropy is:

$$H(E_A) = \tfrac{2n-1}{2n} log_2 \tfrac{1}{(2n-1)/2n} + \tfrac{1}{2n} log_2 \tfrac{1}{1/2n} = 0.0703,$$
$$H(E_B) = 0.0703,$$
$$avg = \tfrac{H(E_A)+H(E_B)}{2} = 0.0703.$$

The cluster entropy is:

$$H(C_1) = \tfrac{2n-1}{2n} log_2 \tfrac{1}{(2n-1)/2n} + \tfrac{1}{2n} log_2 \tfrac{1}{1/2n} = 0.0703,$$
$$H(C_2) = 0.0703,$$
$$avg = \tfrac{H(C_1)+H(C_2)}{2} = 0.0703.$$

Let us observe that in contrast to the diversity and dispersion, the entropy-based measures do capture that the second partitioning is better than the first one, since $0.0703 < 1.0$.

5. EXPERIMENTAL EVALUATION

In this section, we experimentally study the proposed approach on a real dataset. We conducted the experiments on a 2GHz Pentium 4 machine with 1GB RAM. In the rest of this section, we first describe the dataset we use and then present the experiments that test the quality and the efficiency of the proposed technique.

5.1 RealMov Dataset

One of the dataset used in data cleaning research is the movies dataset, available from [34]. It is a real public-domain dataset. In this paper we refer to a processed version of it as 'RealMov'. RealMov contains entities of three types: *movies* (11,453 entities), *studios* (992 entities), and *people* (22,121 entities), which are stored in *movies*, *studios* and *people* tables respectively. The *movies* table contains multiple attributes, such as the title of a movie, the director, the producer, the studio producing the movie, the studio distributing the movie, etc. The *studios* table has attributes such as studio name, the founder of the studio, the year of foundation, etc. The *people* table contains attributes like a person's name, the date of birth, gender, etc. There is also a *cast* table storing all the actors of each movie.

This dataset contains five types of (regular) relationships: *movie_actor*, *movie_director*, *movie_producer*, *producingStudio*, and *distributingStudio*, which map movies to their actors, directors, producers, producing studios and distributing studios respectively. Figure 13 presents a sample graph for RealMov dataset. Relationships *movie_actor*, *movie_director*, and *movie_producer* connect entities of type *movies* to entities of type *people*. Relationships *producingStudio* and *distributingStudio* connect entities of types *movies* and *studios*.

A clean version of this dataset is available, in which all entities and relationships among them are accurately captured. This will allow us to test the quality of various consolidation techniques: by comparing their output against the true situation. Figure 13 shows a sample ARG for RealMov dataset. Each entity is represented as a node, and each relationship as an edge.

5.2 Quality experiments

Typically, two aspects of data cleaning algorithm are evaluated empirically: the quality of their outcome and their

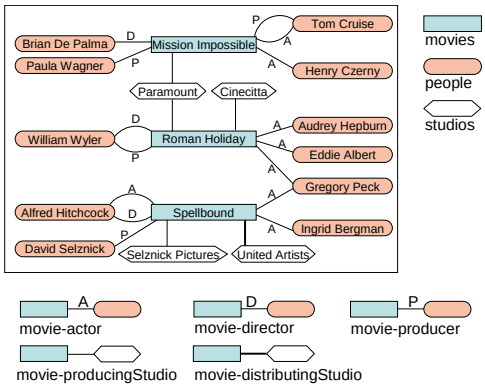


Figure 13: Graph example for movie dataset

efficiency. In this section, we study the quality of our approach on consolidating the representations of *director* entities. Let us note that the true mapping between the movies and their directors is available. Having this knowledge is the advantage of this dataset, since it will allow to compute the quality of various consolidation techniques.

Constructing experiment data. To test our approach, we will use a standard technique, commonly employed by data cleaning practitioners, e.g. in [4]: we introduce uncertainty/errors (in director representations) manually. The uncertainty will be introduced differently in different experiments, as explained next.

Assume that RealMov stores information about $d_1, d_2, \ldots, d_n$ director entities. We first choose randomly a fraction ρ of those directors: all their representations will be made uncertain. Based on the typical degree of uncertainty in real-world dataset [17], we set ρ to either 1%, 5%, 10%, 15%. Suppose that, say, the directors d_1, d_2, …, d_{10} were chosen. We will make uncertain all the representations that refer to them, whereas for the rest of the directors $d_{11}, d_{12}, \ldots, d_n$, all their representations will still uniquely identify the right director.

To achieve that, we group the "uncertain" directors d_1, d_2, …, d_{10} in some fashion, say in groups of two, e.g. $\{d_1, d_2\}$, $\{d_3, d_4\}$, …, $\{d_9, d_{10}\}$. Then, we simulate the desired FBS uncertainty by changing all the representations of directors that belong to one group, such that each representation can refer to all directors in this group – if only the FBS similarity is utilized. For instance, assume that the dataset contains a representation r_1, which could only represent d_1. We modify r_1 such that it will fit the description of both, d_1 and d_2, – but will not fit the description of any other director. Such a constructed dataset is characterized by two types of parameters: ρ and the sizes of those groups.

Baseline methods. To reflect how our approach would compare against FBS techniques, we construct two baseline methods: *Baseline 1* and *Baseline 2*. Recall that our algorithm is applied only to 'tough' cases – after the existing FBS methods have already been used to successfully consolidate many of the representations (e.g., those "certain" representations). We now only test the quality of consolidating those 'tough' cases, which cannot be disambiguated by FBS methods.

Given that FBS cannot be used further to distinguish between the representations, whereas we would like to compare our algorithm against at least simple solutions, we construct our baseline methods as follows.

- *Baseline 1* method creates one cluster per each VCS and then assigns all the VCS's representations to this one cluster. That is, that method does not partition the VCS's further. This naïve method always achieves the ideal dispersion and entity entropy, because entities end up in just one cluster, as they should. However, if the VCS contains the representation of m objects, then the diversity of the cluster will be m, whereas the ideal diversity is 1.
- *Baseline 2* method knows the statistics of how many director groups there are of size 2, of size 3, and so on. Based on this statistics, for a given VCS, it first selects the number of partitions to split this VCS into. It then creates that many clusters, and randomly assigns representations from the VCS to each cluster.

Experiment 1. In this experiment, we set $\rho = 1\%$ and the size of each group of directors to be 2. The results for $\rho = 5\%$, 10% and 15% closely resemble those for $\rho = 1\%$ and thus omitted. For this experiment we also make our algorithm (and Baseline 2) aware that each resulting VCS must be partitioned into exactly two clusters.

Figure 14(c) and 14(d) show the average diversity and dispersion as we vary parameter L . Recall that we consider only L-short paths, or paths of length no greater than L. The results show that additional semantic information, stored in inter-object relationship, improves the quality of object consolidation. They also show that longer paths help improve the quality of the outcome.

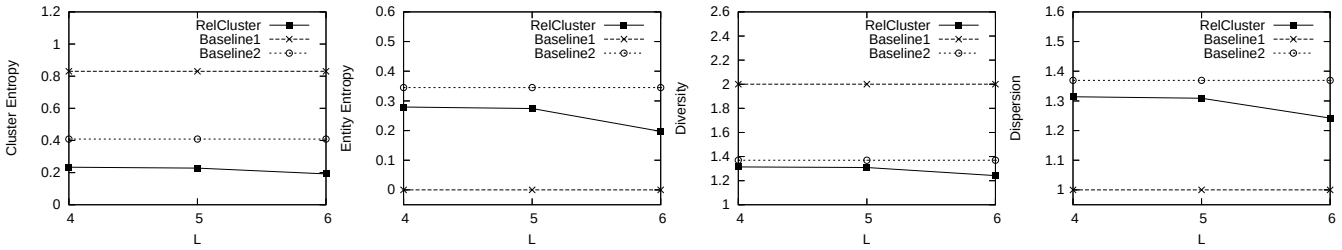
Since Baseline 1 does not partition VCS's at all, each resulting cluster always contains representations of 2 entities, and thus the diversity is always 2. Also, since the nodes of the same entity are always grouped into the same cluster, the entity dispersion is always 1 (the ideal dispersion).

Compared to the diversity and dispersion, the entropy should more properly capture the composition of groups. Figures 14(a) and 14(b) show the average cluster- and entity-entropy, achieved by the consolidation approaches. Recall that the lower the entropy, the better, and that the ideal entropy is zero. The figures look similar to the figures for the diversity and dispersion.

Experiment 2. In this experiment, the size of each director group is not 2 as in the previous experiment, but chosen randomly as 2, 3, or 4 with equal probability. Also our approach now is not aware into how many clusters each VCS should be clustered but rather utilizes threshold τ to decide that, as discussed in Section 4.2.

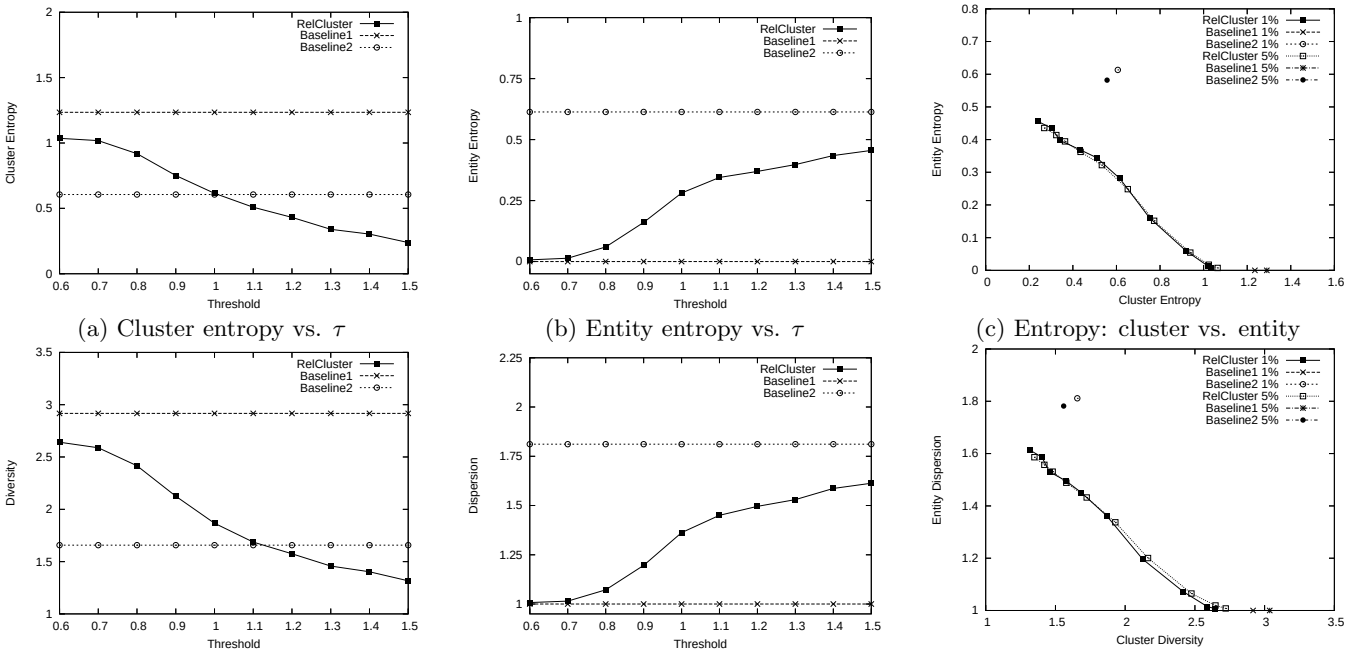
Figure 15 studies the effects of τ on the quality of the output. When τ is small, the normalized cut of most partitions is greater than τ, so the further partitioning is not carried out. Therefore, representations in each VCS are likely to be grouped into a small number of clusters and that is why the results closely resemble those of Baseline 1. On the other hand, large τ leads to creating many clusters for each VCS. This improves the cluster entropy, but the entity entropy becomes worse. So, there is a natural trade-off between the cluster entropy and entity entropy.

Figure 15(c) plots the cluster and entity entropy in one figure. Such a figure is useful for the analysts to pick the right value of τ, such that the desired compromise between the values of cluster- and entity- entropies is achieved. Figures 15(d), 15(e), 15(f) are similar to Figures 15(a), 15(b), 15(c), but for the diversity and dispersion.



(a) Cluster entropy (b) Entity entropy (c) Cluster diversity (d) Entity dispersion

Figure 14: Experiments with various lengths of paths



(a) Cluster entropy vs. τ (b) Entity entropy vs. τ (c) Entropy: cluster vs. entity

(d) Cluster diversity vs. τ (e) Entity dispersion vs. τ (f) Diversity vs. Dispersion

Figure 15: Experiments with various thresholds

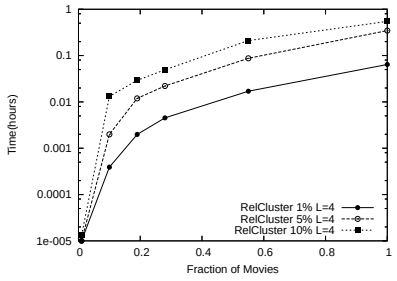


Figure 16: Execution time vs. database size.

5.3 Efficiency

Experiment 3. This experiment tests the efficiency of the proposed approach. Figure 16 shows the execution time of RelCluster as a function of the fraction of movies from RealMov dataset, e.g. 1.0 corresponds to the whole RealMov dataset. The bottleneck of our approach is the algorithm for discovering all L-short simple paths. In [14] we study several optimizations of that algorithm, which improve the performance by 1–2 orders of magnitude. We employ the same optimizations in out implementation of RelCluster.

6. RELATED WORK

Many research challenges have been explored in the context of data cleaning: dealing with missing data, handling erroneous data, record linkage, and so on. The closest to the problem of object consolidation addressed in this paper is the problem of record linkage. The importance of record linkage is underscored by the large number of companies, such as Trillium, Vality, FirstLogic, DataFlux, which have developed domain-specific record linkage solutions.

Researchers have also explored domain-independent techniques, e.g. [1, 10, 13, 19, 23]. Their work can be viewed as addressing two challenges: (1) improving similarity function, as in [3]; and (2) improving efficiency of linkage, as in [4]. Typically, two-level similarity functions are employed to compare two records. First, such a function computes attribute-level similarities by comparing values in the same attributes of two records. Next, the function combines the attribute-level similarity measures to compute the overall similarity of two records. A recent trend has been to employ machine learning techniques, e.g. SVM, to learn the best similarity function for a given domain [3]. Many techniques have been proposed to address the efficiency challenge

as well: e.g. using specialized indexes [4], sortings, etc.

Those domain-independent techniques deal only with attributes. Only one existing approach [17] analyzes relationships in a fashion similar to that proposed in this paper. That approach, and the one proposed in this paper, are part of the *Relationship-based Data Cleaning (RelDC)* project [15] at UCI. However, [17] solves a different data cleaning challenge, called reference disambiguation. That problem is known to be a subproblem of the problem of object consolidation and the approach proposed in [17], in general, cannot be used to solve the problem addressed in this paper. That approach converts the cleaning task to solving a nonlinear programming problem whereas we employ partitioning techniques for data cleaning. Other researchers have also proposed using relationships for cleaning, but in a different fashion. In [1] Ananthakrishna et al. employ similarity of directly linked entities, for the case of hierarchical relationships, to solve the record deduplication challenge. In [18] Lee et al. develop an association-rules mining based method to disambiguate references using similarity of the context attributes: the proposed technique is still an FBS method, but [18] also discusses "concept hierarchies" which are related to relationships. Getoor et al. in [2] use similarity of attributes of directly linked objects, like in [1], for the purpose of object consolidation. However, applying that technique in practice on real-world datasets was identified as future work in that paper. In contrast to the above described techniques, our approach and [17] utilize the CAP hypothesis to automatically discover and analyze relationship chains, thereby establishing a framework that employs systematic relationship analysis for the purpose of cleaning.

7. CONCLUSION

In this paper, we have shown that analysis of inter-object relationships is important for object consolidation and have demonstrated one approach that utilizes relationships for this purpose. Our ongoing work [16] addresses the challenge of automatically adapting the proposed data cleaning techniques to datasets at hand, by learning how to weigh different connections directly from data, in an automated fashion. Solving this challenge, in general, not only makes the approach a plug-and-play solution, but also improves both the accuracy and efficiency of the approach as discussed in [16].

8. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.

[2] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD Workshop*, 2004.

[3] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD'03*.

[4] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD Conf.*, 2003.

[5] P. Christen, T. Churches, and J. X. Zhu. Probabilistic name and address cleaning and standardisation. The Australasian Data Mining Wshp, 2002.

[6] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *SIGKDD*, 2002.

[7] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[8] M. G. Elfeky and V. S. Verykios. On search enhancement of the record linkage process. In *KDD-2003 Wshp on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[9] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.

[10] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[11] H. Garcia-Molina, J. Ullman, and J. Widom. *Database systems: the complete book*. Prentice Hall, 2002.

[12] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB01*.

[13] M. Hernandez and S. Stolfo. The merge/purge prob- lem for large databases. In *SIGMOD*, 1995.

[14] D. Kalashnikov and S. Mehrotra. Exploiting relationships for domain-independent data cleaning. *SIAM SDM*, 2005. ext. ver., `http://www.ics.uci.edu/~dvk/pub/sdm05.pdf`.

[15] D. V. Kalashnikov and S. Mehrotra. RelDC project. `http://www.ics.uci.edu/~dvk/RelDC/`.

[16] D. V. Kalashnikov and S. Mehrotra. Learning importance of relationships for reference disambiguation. *UCI Technical Report RESCUE-04-23*, Dec. 2004. `http://www.ics.uci.edu/~dvk/RelDC/TR/TR-RESCUE-04-23.pdf`.

[17] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM International Conference on Data Mining (SIAM SDM 2005)*, Newport Beach, CA, USA, April 21–23 2005.

[18] M. Lee, W. Hsu, and V. Kothari. Cleaning the spurious links in data. *IEEE Intelligent Systems*, Mar-Apr 2004.

[19] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *SIGKDD*, 2000.

[20] M. Michalowski, S. Thakkar, and C. Knoblock. Exploiting secondary sources for automatic object consolidation. In *KDD-2003 Wshp on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[21] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *SIGMOD Wshp on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[22] M. Neiling and S. Jurk. The object identification framework. In *KDD-2003 Wshp on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[23] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[24] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In Advances in Neural Processing Systems 15. Vancouver, British Columbia:MIT Press, 2002.

[25] D. Quass and P. Starkey. Record linkage for genealogical databases. In *KDD-2003 Wshp on Data Cleaning*, 2003.

[26] L. D. Raedt and et al. Three companions for data mining in first order logic. In *Relational Data Mining*. Springer-Verlag, 2001.

[27] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, 2002.

[28] C. E. Shannon. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[29] J. Shawe-Taylor and N. Cristianni. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[30] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22.

[31] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string tranformation weights for high accuracy object identification. In *SIGKDD*, 2002.

[32] V. Verykios, G.V.Moustakides, and M. Elfeky. A bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12:28–40, 2003.

[33] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *SIGKDD*, 2003.

[34] G. Wiederhold. `www-db.stanford.edu/pub/movies/`.