Exploiting relationships for object consolidation

Zhaoqi Chen Dmitri V. Kalashnikov Sharad Mehrotra

Computer Science Department
University of California, Irvine

ABSTRACT

Researchers in the data mining area frequently have to spend significant portion of their time on preprocessing the data in order to apply their algorithms to real-world datasets. Many real-world datasets are not perfect: they contain missing, erroneous, duplicate data and other 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 ("garbage in, garbage out" principle). Therefore data cleaning techniques should be applied in-advance to the data to ensure high quality of the results.

In this paper we address a data cleaning challenge called *object consolidation*. This challenge arises because often objects in datasets are 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 techniques, our approach analyze not only object features but also inter-objects relationships for object consolidation. Our experiments over real datasets show that analyzing relationships significantly improves the quality of the result.

Categories and Subject Descriptors

H.2.m [**Database Management**]: Miscellaneous – *Data cleaning*;
H.2.8 [**Database Management**]: Database Applications – *Data mining*;
H.2.5 [**Information Systems**]: Heterogeneous Databases;
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Relationship-based data cleaning, record linkage, similarity retrieval

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 datasets being analyzed 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 and companies 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* [28, 10, 31, 34]. 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 Smiith' (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 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 that it seems because of 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'. Also there are existing techniques that are known to group multiple people with common names into one "high-impact" person thus lowering the relative stand of authors with rare names. As a result, the impact of 'John A. Smith' computed on such a dataset can be far from the reality even though a very basic analysis technique is employed to compute it.

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* [32, 12, 16, 1, 27] 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. Another related problem is the problem of *reference disambiguation* [25, 20]. 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, i.e. object consolidation problem is more generic, it does not require a known to be clean list of possible objects.

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 utilize 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 etc, forming chains of relationships among entities. Such knowledge 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 "CAP" hypothesis for historic reasons:

The CAP hypothesis: *(a) 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; (b) if two representations refer to different entities, even though the feature-based similarity between them can be high, the connection between them via the relationships should be weaker comparing with that of the representations referring to the same entity.* □

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

The main contributions of this paper are: (1) developing a systematic approach that in contrast to traditional techniques exploits not only attributes but also inter-object relationships for object consolidation. (2) establishing that exploiting relationships can significantly improve the quality of object consolidation by testing the developed approach over real datasets.

The rest of the paper is organized as follows. Section 2 presents a motivational example. In Section 3, we formalize the problem and introduce the notation that will be used to explain the approach. Section 4 describes the approach and the algorithm being used. The empirical results are presented in Section 5. Section 6 contains the related work, and Section 7 concludes the paper.

2. MOTIVATIONAL EXAMPLES

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

Consider a database about *publications* and *authors*. Assume the publications are represented in the database using the attributes ⟨`id,title,authorRef1`, ..., `authorRefN`⟩

[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.

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

Figure 1: *publication* records

1. ⟨$A1$, **'John Smith'**, 'MIT', 'GradStudent',...⟩
2. ⟨$A2$, **'John Z. Smith'**, 'CMU', 'Faculty', Research(...), ...⟩
3. ⟨$A3$, **'John Smith'**, 'Stanford', 'Faculty', Teaching(...), ...⟩
4. ⟨$A4$, 'Alan White', 'MIT', 'Professor',...⟩
5. ⟨$A5$, 'Mike Black', 'NEC', Project(...),...⟩
6. ⟨$A6$, 'Tom Grey', 'Intel', Lab(...), ...⟩
7. ⟨$A7$, 'Kate Red', 'Stanford'⟩

Figure 2: *author* records

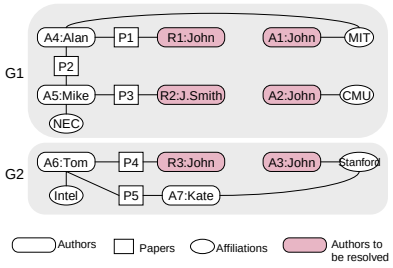


Figure 3: Graph for toy database.

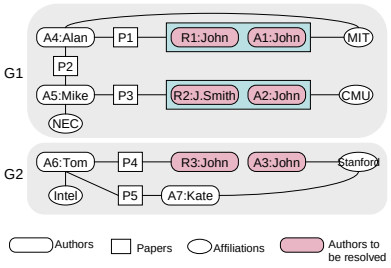


Figure 4: Adding context info.

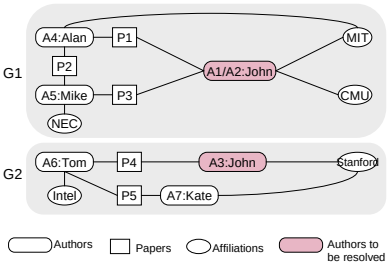


Figure 5: Adding relshp analysis.

and the information about authors is stored in the form ⟨`id,authorName,affiliation, misc1, ..., miscM`⟩. Here `misc` attributes can store some auxiliary information about authors, such as graduate school, advisor, etc. Consider a toy database consisting of the *author* and *publication* records shown in Figures 1 and 2.

We assume that in this database all types of entities, except for entities of type *author*, can be unambiguously determined based on their representations. That is, we are confident that organization called 'MIT' in $A1$ is the same as 'MIT' in $A4$. The challenge in this case is to correctly consolidate only *author* representations. For instance, author identified as 'John Smith' in $P1$ might be the same person who wrote $P3$ and the same person identified as $A1$, or, alternatively, those three representations might be for three different people. Also for that matter $A1$ and $A2$ can be duplicate records for the same person, if, say, 'John Smith' was a graduate student at MIT at some time and then he went on to become a faculty at CMU.

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 match most of the representations for our toy database. 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 $A3$.

A few of most recent data cleaning techniques are capable of using context to improve the quality of cleaning. There might be additional *context* information 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 $A2$ or $A3$. As another example, if we know that $A2$ has written several papers before with titles similar to that of $P3$ we can infer that 'J. Smith' in $P3$ refers to $A2$.

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

First, to handle author 'John Smith' in $P1$, we observe that his co-author 'Alan White' has also written $P2$ 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'. Also notice that since we have determined before that 'John Smith' in $P1$ refers to $A1$ and 'J. Smith' in $P2$ to $A2$, the evidence suggests that $A1$ and $A2$ are duplicate records for the same author – the fact that was not captured by feature-based similarity deduplication of *author* records!

Second, to handle author 'John Smith' in $P4$ in order to decide whether it refers to $A1$, $A2$, or $A3$, we observe that his coauthor, 'Tom Grey', has a paper $P5$ with 'Kate Red' who is from 'Stanford'. The author $A3$ is also from 'Stanford' and thus we were able to establish a connection between 'John Smith' in $P4$ and $A3$. Given that there are no connection between 'John Smith' in $P4$ and $A1$ or $A2$, we might decide that 'John Smith' in $P4$ refers to $A3$.

At first glance, the analysis above might seem to be domain-specific. However, 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).

Figure 3 shows a possible graph representation of the publication database given above. For instance, since author $A1$ is affiliated with 'MIT' there is an edge between them that corresponds to this relationship. If for certain entities their representations uniquely identify them, then only one node is created per entity. For example, for the toy database, representations for affiliations uniquely identify each entity, therefore only one node was created for 'MIT'. For the rest of the representations, which might need to be consolidated, a node is created per representation. For example, each 'John Smith' has a separate node in Figure 3.

Figure 4 is similar to Figure 3, it shows that based on the attributes derived from the context it was determined that $R1$ should be consolidated with $A1$ and $R2$ with $A2$. Figure 4 shows that $R1$ and $A1$ are grouped into one node, and the same for $R2$ and $A2$.

The analysis above can be viewed as application of the CAP hypothesis to the toy database represented by 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 (i.e., *relationship chain* or *connection*)

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

which connects $R1$ and $R2$. The second observation we made regarding disambiguation of 'John Smith' in $P4$ was based on the presence of path

$R3 \rightarrow P4 \rightarrow A6 \rightarrow P5 \rightarrow A7 \rightarrow$ 'Stanford' $\rightarrow A3$

which connects $R3$ and $A3$.

Figure 5 shows the resulting graph which reflects how references should be consolidated based on the above analysis. In this figure $R1$, $R2$, $A1$, and $A2$ are grouped together which indicates that based on the analysis those representations are likely to refer to the same person. Similarly $R3$ and $A3$ are grouped as well. Notice 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 in this example and a similar analysis will need to measure the "strength" of the connections that exists between various entities.

Therefore the generic approach for object consolidation can consists of the following steps. First it identifies the representations of potentially the same entity, then it discovers *connections* between these representations and measures the connection strength between them to obtain the evidence that they represent the same entity. The algorithm then employs one of the existing partitioning algorithms to group the representations into clusters such that the connections between the nodes in the same cluster are strong and between clusters are weak.

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 extensive testing of our approach over real data to establish that exploiting relationships significantly improves the data quality. Before we develop our solution, we first develop notation and concepts needed to explain our approach in Section 3.

3. PROBLEM FORMALIZATION

3.1 Notation

Let $\mathcal{D}$ be the database being processed. Let $O = \{o_1, o_2, \ldots, o_{|O|}\}$ be the set of all entities (or, *objects*) in $\mathcal{D}$. Entities here have the same meaning as in the E/R model. We use entities only semantically: the algorithm does not necessarily know about them or even the number of them. 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 `authorRef` representations consisted of only one attribute and were in the form $\langle$*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 consolidation algorithms. The **goal** is to group representations in X into the set 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 while no two representations from two different clusters refer to the same entity.[2] That is for any two representations x and y from cluster C it should follow that $d[x] = d[y]$, and 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 should be put into the same group as representation x_i: $C[x_i] = \{x_j \in X : d[x_j] = d[x_i]\}$. 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: $S[x_i] = \{x_j : FBS(x_j, x_i) > \tau\}$, where FBS denote a feature-based similarity function and τ is some threshold. We assume that $C[x_i] \subseteq S[x_i]$.

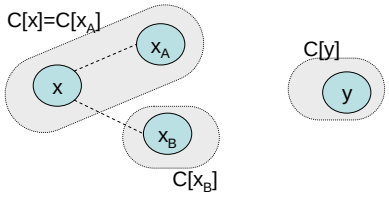


Figure 6: Similarity edges.

Let us illustrate those concept by considering a database which contains three representations of people 'J. Smith' (x), 'J. A. Smith' (x_A), 'J. B. Smith' (x_B), and 'Alan White' (y). Let us assume those representation are not misspelled and therefore x_A and x_B cannot be representations of the same person. Assume x and x_A are representations of one

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

person, x_B is of another person, and y is of a third person. Then
$C[x] = C[x_A] = \{x, x_A\}$, $C[x_B] = \{x_B\}$,
$S[x] = \{x, x_A, x_B\}$, $S[x_A] = \{x, x_A\}$, $S[x_B] = \{x, x_B\}$
$C[y] = S[y] = \{y\}$.

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 representations can refer to the same entity based on feature-based similarity.

In general such a *virtual similarity graph* can be created for all the representations in $\mathcal{D}$. Typically it is composed of multiple *virtual connected subgraphs (VCS)*, e.g. 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 would allow us to cluster references in a "one VCS at a time" fashion. We will use notation $S^*[x_i]$ to refer to all of the representations that belong to the same VCS as x_i. Notice that $C[x_i] \subseteq S[x_i] \subseteq S^*[x_i]$. For instance, for the case demonstrated in Figure 6, we have $S^*[x] = S^*[x_A] = S^*[x_B] = \{x, x_A, x_B\}$, $S^*[y] = \{y\}$.

3.2 The Attributed Relational Graph

Our 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. Normally ARGs are employed to represent entities (nodes) connected via relationships (edges). The graph is called *attributed* because attributes can be associated with both the edges and nodes of such a graph. Our ARG has some specifics as discussed 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 shows the graph that would result by applying this procedure to the toy database. Notice that the way our approach creates nodes closely resembles the way they are typically created in ARGs – to represent distinct entities.

Edges. Our approach considers 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 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.

[3]We will concentrate primarily on binary relationships. Multiway relationships are rare and most of them can be converted to binary relationships [13]. Most of the design models/tools only deal with binary relationships, for instance ODL (Object Definition Language) supports only binary relationships.

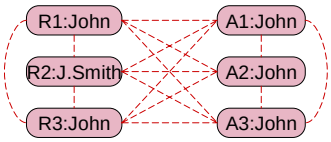


Figure 7: Virtual connected subgraph

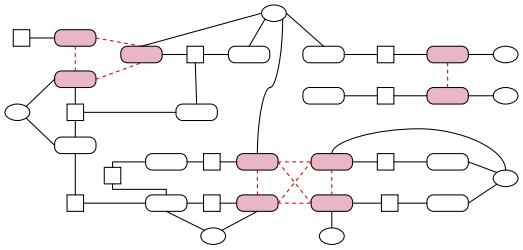


Figure 8: Graph Example

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 one illustrated in Figure 8. Notice how similarity edges define three distinct VCS's in this example.

3.3 Connection Strength

As mentioned before, our approach consolidates representations based on the CAP hypothesis that was discussed in Section 1. For that it utilizes the notion of connection strength between two representations x and y (denoted $c(x, y)$) which captures how strongly x and y are connected to each other through relationships. Many different approaches can be used to compute $c(x, y)$, one of them will be discussed in Section 4.1.

4. THE APPROACH

We now have developed all the concepts and notation needed to explain our approach for object consolidation. As was mentioned, that task of consolidation can be trivial for some of the representation, therefore our solution will concentrate mostly on consolidating the non-trivial cases. The approach exploits both features and relationships for consolidation and outputs the resulting clustering as it outcome. The approach consolidates representations using the following steps:

1. **Construct the ARG and identify all VCS's.** The first step of our approach is to construct the ARG on

which the algorithm will work. We assume feature-based similarity is used in constructing such a graph. This step has been explained in Section 3.2.

2. **Compute connection strengths** Given the constructed graph, compute the connection strengths among the nodes within each VCS.
3. **Partition each VCS**. Use one of the partitioning algorithms to partition each VCS into clusters based on the connection strengths computed in Step 2.

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

4.1 Computing Connection Strength

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

4.1.1 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. A path is *L-short* if its length is no greater than parameter L. A path is *simple* if all its nodes are distinct.

4.1.2 Measuring the connection strength.

Many models for measuring $c(u, v)$ can be derived from [11, 41, 23]. We will use a model which computes $c(u, v)$ as the sum $\sum_{p \in \mathcal{P}_L(u,v)} c(p)$ of the connection strength $c(p)$ of each path p in $\mathcal{P}_L(u, v)$. It will compute the connection strength $c(p)$ of each individual path p in a specific way, which we motivate next.

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

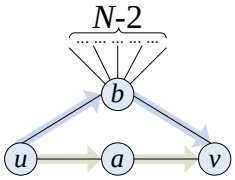


Figure 9: Motivating $c(p)$.

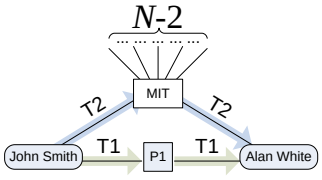


Figure 10: $c(p)$.

Figure 11: Experiments

paths (or connections) between nodes u and v: $p_a = u \rightarrow a \rightarrow v$ and $p_b = u \rightarrow b \rightarrow v$. 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 things, 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 things 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 $c(p_a) > c(p_b)$.

We compute $c(p)$ as follows. Assume first path p contains 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

$$c(p) = w_1^{n_1} w_2^{n_2} \times \cdots \times w_m^{n_m}. \quad (1)$$

Even though the value of those weights can be learned automatically from the data being processed [22], we do not consider such techniques in this paper and instead assume that those weights are assigned by a domain analyst. For instance, 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 $c(p_a) = w_1^2 = \frac{1}{4}$ and $c(p_b) = w_2^2 = \frac{1}{100}$.

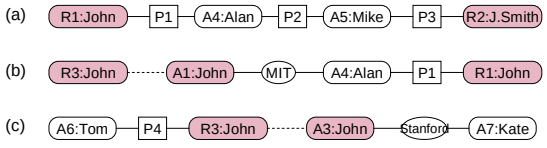


Figure 12: Paths with/without similarity edges.

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 entity and it has a weight associated with it which reflects the degree of similarity between x and y based on their features. We will refer to this weight as the FBS weight. Notice, a path that contain 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 exist. However, the existence of paths in Figures 12(b) depends on whether $R3$ and $A1$ refer to the same entity or not, and the same for $R3$ and $A3$ in Figure 12(c).

The simplest solution is not to consider paths that contain similarity edges at all. Another solution we use is to associate the same very small *base* weight w_ε for all similarity edges. Then compute the total weight of a similarity edge as a product of its FBS weight and the base weight. 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

$$c(p) = w_1^{n_1} w_2^{n_2} \times \cdots \times w_m^{n_m} v_1 v_2 \times \cdots \times v_k. \quad (2)$$

The total connection strength between nodes u and v is computed as the sum of connection strengths of paths in $\mathcal{P}_L(u, v)$:

$$c(u, v) = \sum_{p \in \mathcal{P}_L(u,v)} c(p). \quad (3)$$

4.2 Consolidating objects by partitioning VCS's

Choosing a partitioning algorithm. To consolidate objects we can use one of the existing min-cut algorithms to partition each VCS separately. The general min-cut problem is, given a (weighted) graph $G = (V, E)$, 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 (the *cut*, weight of $\{(u, v) \in E : u \in V_1; v \in V_2\}$) is minimized. Each partition then can be repeatedly subdivided further, until some desired properties are satisfied.

We use the *normalized cut (NCut)* algorithm proposed in [38]. As the name suggests, that algorithm instead of using the regular cut utilizes the "normalized cut" which is defined as a fraction of the total edge connections to all the nodes in the graph:

$$Ncut(V_1, V_2) = \frac{cut(V_1, V_2)}{assoc(V_1, V)} + \frac{cut(V_1, V_2)}{assoc(V_2, V)}.$$

Here $cut(V_1, V_2)$ has the standard definition:

$$cut(V_1, V_2) = \sum_{u \in V_1, v \in V_2} w(u, v),$$

and $assoc(V_1, V)$ is the total connection from nodes in V_1 to all nodes in the graph, which is computed as

$$assoc(V_1, V) = \sum_{u \in V_1, v \in V} w(u, v).$$

Measure $assoc(V_2, V)$ is similarly defined.

The connection within the two clusters is measured by the "normalized association":

$$Nassoc(V_1, V_2) = \frac{assoc(V_1, V_1)}{assoc(V_1, V)} + \frac{assoc(V_2, V_2)}{assoc(V_2, V)}.$$

The authors have proved, that the normalized cut and the normalized association are related:

$$Ncut(V_1, V_2) = 2 - Nassoc(V_1, V_2).$$

Hence, partition criteria that we would like to see in our grouping algorithm, minimizing the disassociation between the groups and maximizing the association within the groups, are in fact identical and can be satisfied simultaneously. That was the reason for choosing this algorithm for partitioning.

Defining weights for partitioning. A min-cut algorithm uses weights $w(u, v)$ between nodes for partitioning. We define those weights as follows: if there is a similarity edge between u and v then $w(u, v) = c(u, v)$, otherwise $w(u, v) = 0$. Notice that potentially, when defining $w(u, v)$, we could have considered not only $c(u, v)$ but also the degree of feature-based similarity between u and v.

Further partitioning. Since we do not know the actual number of entities represented by a given VCS, we do not know the number of clusters we should output. So after the VCS is partitioned into two parts, we need to decide whether to actually separate the nodes or not. We achieve that by comparing the value of the normalized cut, c, with a certain threshold τ. If $c > \tau$, the components are still well inter-connected and we assume that the VCS should not be divided further into two clusters. That means we assume all of the representations in this VCS refer to the same real-world entity and hence should be grouped together. On the other hand, if $c < \tau$, we 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 the outcome

The goal of object consolidation algorithms is to accurately group the representations of entities. However, consolidation algorithms 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. Also we want each cluster to contain representations of just one entity. The *diversity* of a given cluster captures the number of distinct entities to which representations in this cluster refer. Similarly, the lesser the diversity the better and the ideal diversity is 1 for a given cluster.

Entropy. 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 being 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 would be two clusters $C_1 = \{a_1, a_2, \ldots, a_{2n}\}$ and $C_2 = \{b_1, b_2, \ldots, b_{2n}\}$. The algorithm however can make mistakes and the resulting two clusters might be:

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

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

1. $C_1 = \{a_1, a_2, \ldots, a_{2n-1}, b_{2n}\}$
2. $C_2 = \{b_1, b_2, \ldots, b_{2n-1}, a_{2n}\}$

that is, just one is misassigned for each cluster. Obviously, the latter answer is better. But in both situations, since the representations of both E_A and E_B are scattered among 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 can not distinguish the two situations in this case.

There is a known metric, called *entropy*, which was first introduced by Shannon in [37], that can be utilized to better capture the above situations. 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 respective probabilities $p(x_1), p(x_2), \ldots, p(x_n)$, where $p(x_1)+p(x_2)+\cdots+p(x_n) = 1$. Then

$$Entropy(X) = H(X) = \sum_{\substack{i=1 \\ p(x_i)\neq 0}}^{n} p(x_i)\, log_2 \frac{1}{p(x_i)}$$

Entropy $H(X)$ attains its minimum value, $H(X) = 0$, when there exists some i such that $p(x_i) = 1$, and we are certain that x_i is the value of variable X and thus there is no uncertainty associated with X. On the other hand, $H(X)$ attains the maximum value when all the values are equally likely, in which case $H(X) = log_2\, n$.

Entropy of Entities: 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. Note, those n clusters are not all clusters, but only those to which the representations of E were assigned. Assume that m_1 representations were assigned to cluster C_1, m_2 to C_2 and so on, where $m_1 + m_2 + \cdots + m_n = m$, $m_i \neq 0$. To measure the spread of representations of E over clusters we can consider which fraction of all representations of E were assigned to each cluster C_i $(i = 1, 2, \ldots, n)$: $p_i = \frac{m_i}{m}$. Then, using those fractions we can employ entropy-like measure to quantify the spread of the entity's representations:

$$H(E) = \sum_{i=1}^{n} p_i\, log_2 \frac{1}{p_i}$$

Notice that dispersion in this case would always return n. The lower the value of entropy of an entity the better, the ideal value is 0.

Entropy of Clusters: Similarly we can define the Entropy of Clusters. Assume that a certain cluster C was assigned m representations(corresponding to n entities) by the algorithm, m_1 of which are for some entity E_1, m_2 for E_2, ..., m_n for E_n, such that $m_1 + m_2 + \cdots + m_n = m$, $m_i \neq 0$. We can similarly consider fractions $p_i = \frac{m_i}{m}$ and then use them in an entropy-like measure to quantify the spread of the cluster's representations:

$$H(C) = \sum_{i=1}^{n} p_i\, log_2 \frac{1}{p_i}$$

The diversity in this case would always return n. The lower the value of entropy of a cluster the better, the ideal value is 0.

If we go back to the example above and assume $n = 10$, in the first situation, the entropy of entities is:
$H(E_A) = \frac{n}{2n} log_2 \frac{1}{n/2n} + \frac{n}{2n} log_2 \frac{1}{n/2n} = 1$,
$H(E_B) = 1$, and
$AvgEntityEntropy = \frac{H(E_A)+H(E_B)}{2} = 1$.

The entropy of clusters is:
$H(C_1) = \frac{n}{2n} log_2 \frac{1}{n/2n} + \frac{n}{2n} log_2 \frac{1}{n/2n} = 1$,
$H(C_2) = 1$, and
$AvgClusterEntropy = \frac{H(C_1)+H(C_2)}{2} = 1$.

For the second situation, the entropy of entities is:
$H(E_A) = \frac{2n-1}{2n} log_2 \frac{1}{(2n-1)/2n} + \frac{1}{2n} log_2 \frac{1}{1/2n} = 0.0703$,
$H(E_B) = 0.0703$, and
$AvgEntityEntropy = \frac{H(E_A)+H(E_B)}{2} = 0.0703$.

The entropy of clusters is:
$H(C_1) = \frac{2n-1}{2n} log_2 \frac{1}{(2n-1)/2n} + \frac{1}{2n} log_2 \frac{1}{1/2n} = 0.0703$,
$H(C_2) = 0.0703$, and
$AvgClusterEntropy = \frac{H(C_1)+H(C_2)}{2} = 0.0703$.

Notice that the entropy is able to capture that the second partitioning is better than the first one.

5. EXPERIMENTAL RESULTS

In this section we experimentally study the approach we proposed using a real dataset.

5.1 RealMov Dataset

RealMov is a real public-domain movies dataset available from [42]. It 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

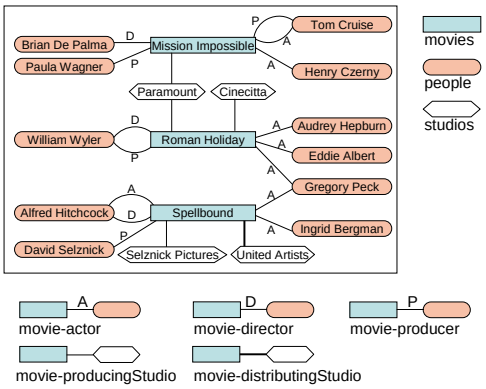


Figure 13: Graph example for movie dataset

for RealMov dataset. Relationships *movie_actor*, *movie_director*, and *movie_producer* connect entities of type *movies* to enti-

ties of type *people*. Relationships *producingStudio* and *distributingStudio* connect entities of type *movies* and *studios*.

A clean version of RealMov dataset is available which allow as to use it as a validation dataset. That is, for this dataset we know all entities and relationships exactly and we know the true matches for all representations. This would allow us to test the quality of various consolidation algorithms by comparing their output against the true situation. Figure 13 shows a sample ARG for RealMov dataset. Each entity is represented by a node and each relationship by an edge.

5.2 Accuracy experiments

Typically two key aspects of data cleaning algorithm are analyzed empirically: the quality of their outcome and the efficiency of those algorithms. In this section we will study the quality of our approach on consolidating the representations of *director* entities.

Notice, as we mentioned, the knowledge of which director representation maps to which director is available in this dataset. That is, there is no uncertainty in how to map director representations to directors and thus how to consolidate director entities. Having this knowledge is the advantage of this dataset since it will allow us to use it for validation.

Constructing experiment data. To test our approach we will use one of the standard techniques commonly employed by data cleaning practitioners [4]: we will introduce uncertainty in director representations manually. This uncertainty will be introduced differently in different experiments.

Assume the RealMov stores information about $d_1, d_2, \ldots, d_n$ director entities. We first choose a fraction ρ of those directors: representations of them will be made uncertain. That is, we may first choose $d_1, d_2, \ldots, d_{10}$ "uncertain "directors, for the rest of the directors $d_{11}, d_{12}, \ldots, d_n$ their representations will uniquely identify the right director. Based on the typical degree of uncertainty in real-world dataset, we set ρ to be ρ=1%, 5%, 10%, 15%. Next we group those "uncertain" directors in some fashion, say in groups of two, e.g. $\{d_1, d_2\}, \{d_3, d_4\}, \ldots, \{d_9, d_{10}\}$. Then we simulate the desired FBS uncertainty by changing all representations of directors that belong to one group such that such a representation can refer to all directors in this group based on FBS similarity. For instance, assume the dataset contained a representation r_1 which could only represent d_1. We modify r_1 such that now it fits the description of both d_1 and d_2 – but does not fit the description of anybody else. Such a constructed dataset is characterized by two types of parameters: ρ and sizes of those groups.

Baseline methods. To reflect how our approach would compare with FBS methods we construct two baseline methods: *Baseline 1* and *Baseline 2*. Notice, at the moment of applying our algorithm, the existing FBS methods already have been used to successfully consolidate many of the representations (those "certain" representations). We now only test the quality of consolidating of those representations that, based on FBS similarity, are still "uncertain". 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 those baseline methods as follows.

Baseline 1 creates a cluster per each resulting 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 method always achieves the ideal dispersion and entropy of entities: entities end up in just one cluster as they should.

Baseline 2 knows home many director groups are there of size 2, of size 3, ..., e.g., it knows the probability of a group that have size 2, 3, ... Based on those probabilities, for a given VCS, it first selects into how many partitions it wants to split this VCS. 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.

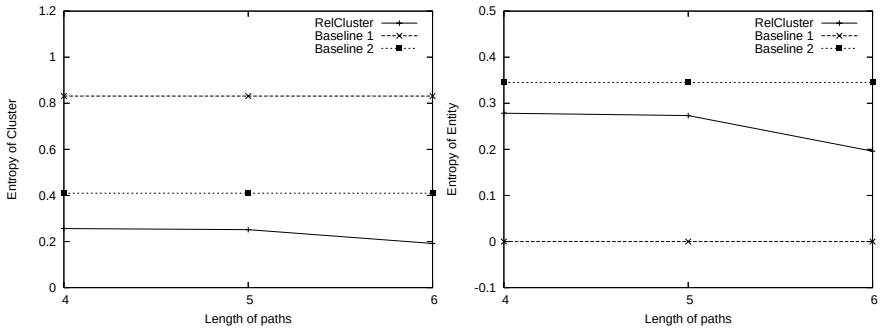
Figure 14(c) and (d) show the average diversity and dispersion as we vary parameter L . Recall that we consider only L-short paths, or paths of length no greater than L. The results of our approach ('RelCluster') shows that additional semantic information stored in inter-object relationship improves the quality of object consolidation. 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; since the nodes of the same entity are always grouped into the same cluster, the entity dispersion is always 1 (the ideal dispersion).

In contrast to the diversity and dispersion, the entropy more properly captures the composition of groups. Figures 14(a) and (b) show the average entropy of clusters and entities achieved by the consolidation approaches. Recall that the lower the entropy the better and that the ideal entropy is 0. The figures look similar to the figures for the diversity and dispersion.

Those figures also show that analyzing longer paths do help increase the quality of outcome.

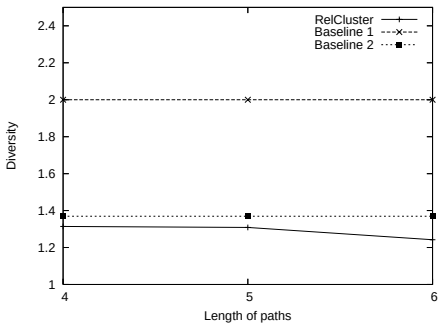
Experiment 2. In this experiment, the size of each director group is not 2 as in the previous experiment, but chosen randomly as 2, 3, or 4 with equal probability. Also our approach now is not aware into how many clusters each VCS should be clustered but rather utilizes threshold τ to decide that, as 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 will be greater than τ, so the partitioning will not be carried out. Therefore, representations in each VCS are likely to be grouped into a small number of cluster 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

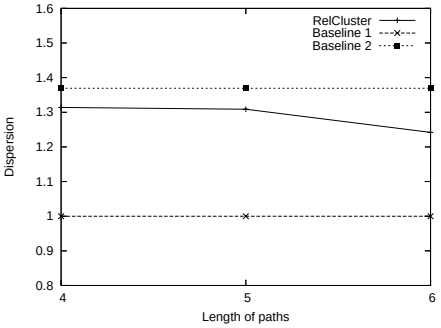


(a)Entropy of Clusters

(b) Entropy of Entities

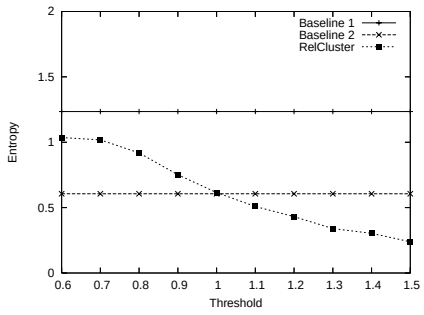


(c)Cluster Diversity

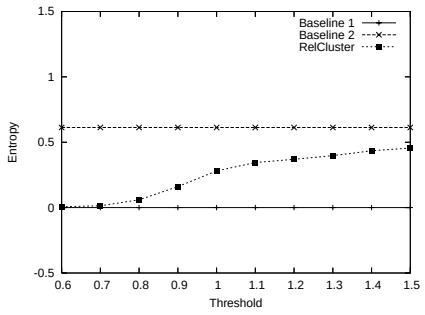


(d) Entity Dispersion

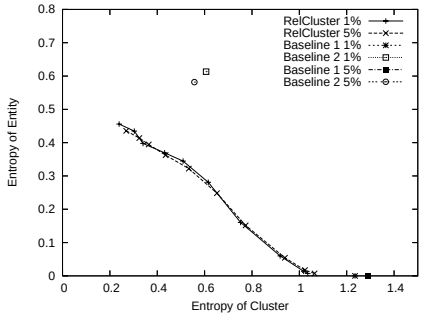
Figure 14: Experiments with various lengths of paths



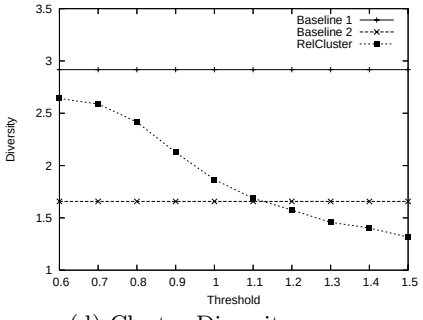
(a) Entropy of Clusters vs. τ



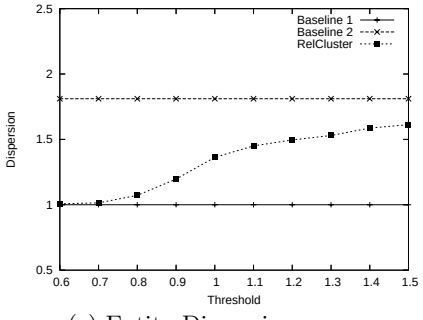
(b) Entropy of Entities vs. τ



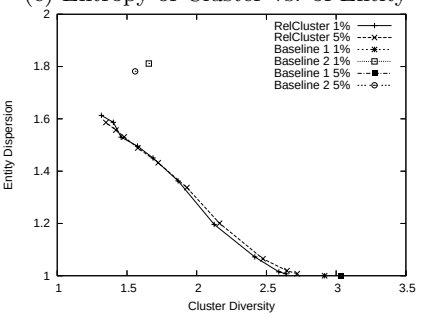
(c) Entropy of Cluster vs. of Entity



(d) Cluster Diversity vs. τ



(e) Entity Dispersion vs. τ



(f) Cluster Diversity vs. Entity Dispersion

Figure 15: Experiments with various thresholds

VCS. This improves the entropy of clusters, but the entropy of entities becomes worse. So, there is a natural trade-off between the entropy of clusters and entropy of entities, This trade-off is similar in nature to the precision/recall trade-off in record linkage.

Figure 15(c) plots the entropy of clusters and of entities in one figure. The analyst supervising the object consolidation process can apply our approach on training datasets and then use a figure similar to Figure 15(c) to choose a good value for τ. Figures 15 (d)—(f) are similar to Figures 15

(a)—(c) but for the diversity and dispersion.

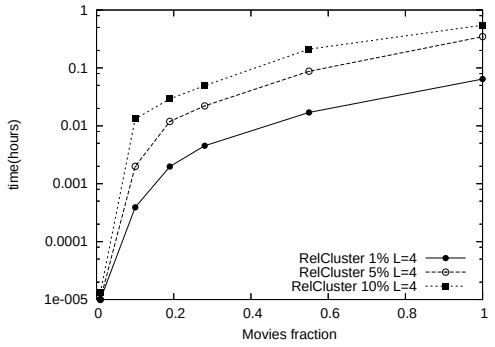


Figure 16: Execution time.

5.3 Efficiency experiments

Experiment 3 In addition to the accuracy, we also test the efficiency of our 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 [20] we study several optimizations of that algorithm which improve the performance by 1–2 orders of magnitude.

6. RELATED WORK

Many research challenges have been explored in the context of data cleaning in the literature: 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. [32, 12, 16, 1, 27]. 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 [23] analyzes relationships in a fashion similar to that proposed in this paper. However, [23] 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 [23], 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 [25] 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 [25] 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, the challenge of 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 [23] 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 demonstrated one approach that utilizes relationships for this purpose. As future work we plan to apply similar techniques for the purpose of record linkage and look into utilizing time and space attributes, which might be present on nodes of discovered paths, for the purpose of cleaning. Our ongoing work [22] addresses the challenge of automatically adapting of a similar approach to datasets at hand by learning how to weigh different connections directly from the data. Solving this challenge, in general, not only makes the approach to be a plug-and-play solution but also can improve the accuracy as well as efficiency of the approach as discussed in [22].

8. REFERENCES

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

[2] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 11–18, Paris,France, 2004.

[3] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, 2003.

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

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

[6] W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *Proc. of ACM SIGKDD Conf.*, 2000.

[7] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proc. of ACM SIGMOD Conf.*, 1998.

[8] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. IIWeb Workshop, 2003.

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

[10] M. G. Elfeky and V. S. Verykios. On search enhancement of the record linkage process. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 31–33, Washington, DC, 2003.

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

[12] I. Fellegi and A. Sunter. A theory for record linkage. *J. of Amer. Stat. Assoc.*, 64(328):1183–1210, 1969.

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

[14] L. Getoor. Multi-relational data mining using probabilistic relational models: research summary. In *Proc. of MRDM Workshop*, 2001.

[15] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *Proc. of VLDB Conf.*, 2001.

[16] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proc. of SIGMOD*, 1995.

[17] M. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406), 1989.

[18] M. Jaro. Probabilistic linkage of large public health data files. *Statistics in Medicine*, 14(5-7), Mar-Apr 1995.

[19] L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *Proc. of DASFAA Conf.*, 2003.

[20] D. Kalashnikov and S. Mehrotra. Exploiting relationships for domain-independent data cleaning. *Extended Version of SIAM Data Mining 2005 publication*, `http://www.ics.uci.edu/~dvk/pub/sdm05.pdf`.

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

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

[23] 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.

[24] e. a. L. De Raedt. Three companions for data mining in first order logic. In *Dzeroski, S. and Lavrac, N., ed. Relational Data Mining*. Springer-Verlag, 2001.

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

[26] M. Lee, H. Lu, T. Ling, and Y. Ko. Cleansing data for mining and warehouse. In *Proc. of DEXA Conf.*, 1999.

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

[28] M. Michalowski, S. Thakkar, and C. Knoblock. Exploiting secondary sources for automatic object consolidation. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 34–36, Washington, DC, 2003.

[29] A. E. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proc. of SIGKDD Conf.*, 1996.

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

[31] M. Neiling and S. Jurk. The object identification framework. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 37–39, Washington, DC, 2003.

[32] Newcombe, Kennedy, Axford, and James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[33] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In Advances in Neural Processing Systems 15, 2002.

[34] D. Quass and P. Starkey. Record linkage for genealogical databases. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 40–42, Washington, DC, 2003.

[35] E. Ristad and P. Yianilos. Learning string edit distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(5):522–532, May 1998.

[36] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proc. of ACM SIGKDD Conf.*, 2002.

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

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

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

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

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

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

[43] W. Winkler. The state of record linkage and current research problems. In *U.S. Bureau of Census, TR99*.

[44] W. E. Winkler. Advanced methods for record linkage. In *U.S. Bureau of Census*, 1994.