Int J Digit Libr (2007) 6:313–326
DOI 10.1007/s00799-007-0014-9

REGULAR PAPER

Practical maintenance of evolving metadata for digital preservation: algorithmic solution and system support

Dongwon Lee

Published online: 3 May 2007

Abstract Metadata (i.e., data describing about data) of digital objects plays an important role in digital libraries and archives, and thus its quality needs to be maintained well. However, as digital objects evolve over time, their associated metadata evolves as well, causing a consistency issue. Since various functionalities of applications containing digital objects (e.g., digital library, public image repository) are based on metadata, evolving metadata directly affects the quality of such applications. To make matters worse, modern data applications are often large-scale (having millions of digital objects) and are constructed by software agents or crawlers (thus often having automatically populated and erroneous metadata). In such an environment, it is challenging to quickly and accurately identify evolving metadata and fix them (if needed) while applications keep running. Despite the importance and implications of the problem, the conventional solutions have been very limited. Most of existing metadata-related approaches either focus on the model and semantics of metadata, or simply keep authority file of some sort for evolving metadata, and never fully exploit its potential usage from the system point of view. On the other hand, the question that we raise in this paper is "when millions of digital objects and their metadata are given, (1) how to quickly identify evolving metadata in various context? and (2) once the evolving metadata are identified, how to incorporate them into the system?" The significance of this paper is that we investigate scalable algorithmic solution toward the identification of evolving metadata and emphasize the role of "systems" for maintenance, and argue that "systems" must keep track of metadata changes pro-actively, and leverage on the learned knowledge in their various services.

D. Lee (✉)
The Pennsylvania State University, University Park, PA, USA
e-mail: dongwon@psu.edu

Keywords Digital preservation · Evolving metadata

1 Introduction

Various digital objects and their associated metadata (i.e., data describing about data) are currently stored and maintained in many data applications. In particular, in this paper, we use the *Digital Libraries and Archives* (hereafter **DL**) as the exemplar data application that stores digital objects (e.g., bibliographic data) and their metadata. In this context, the digital objects range from traditional digital documents (e.g., articles, preprints, technical reports, books, thesis, web pages) to multimedia objects (e.g., images, audio/video files, visualization models), to miscellaneous data sets (e.g., gene data, computer programs, learning objects, administrative records, simulation models). Similarly, metadata ranges from *descriptive* metadata (e.g., description, creator, title, subject) to *administrative* metadata (e.g., access control, creation/deposit date) to *structural* metadata (e.g., file name, type, image resolution) [34]. Since DLs heavily use metadata for various functionalities such as searching and browsing, their quality is directly dependent on that of metadata. As digital objects evolve over time, however, their associated metadata evolves as well, and its accuracy diminishes gradually.

For instance, due to data-entry errors, a DL may contain multiple instances of the same video files with slightly-different metadata, or since a scholar changes her last name due to marriage (e.g., name authority problem [23]), her publications may appear under multiple name variants. Moreover, metadata format (or schema) may also change over time because the adopted metadata standards may change. In order to achieve the long-term integrity of any DLs, therefore, it is important to maintain the metadata of digital objects consistent and up-to-date. When metadata changes over time,

often, it is not only semantically valid but also unavoidable (e.g., author name change due to marriage). Note that the problem of the maintenance of "*evolving metadata*" is an orthogonal issue from: (1) which metadata standard to use (e.g., Dublin Core [13] or Library of Congress core metadata elements [34]); and (2) which persistent ID proposal to use (e.g., DOI [37] or PURL [33]). Therefore, to facilitate the presentation, through the rest of the paper, we will use the notation of Dublin Core as the metadata format. In particular, in the examples, we will focus on the descriptive metadata of Table 1 (taken from [13]).[1] Let us first informally define the evolving metadata.

Definition 1 (Evolving Metadata) When conflicting metadata for a single digital objects occurs, when the value of metadata changes over time, or when the value of metadata becomes obsolete, such a metadata is said to be "**evolving**."

1.1 Motivation

Let us demonstrate examples of evolving metadata drawn from real DLs as the motivation.

1. **Case of** `<creator>` **metadata.** The `<creator>` metadata describes a person, an organization, or a service who is primarily responsible for making the content of the digital object. For instance, in scientific publication DLs (e.g., CiteSeer [29], e-Print arXiv [2], eBizSearch [39] or DBLP [5]), digital objects are scientific articles and `<creator>` metadata is the authors of those articles. That is, the famous article "`As We May Think, V. Bush, The Atlantic Monthly, 1945`" may have "V. Bush" as `<creator>` metadata. Since users often browse or search articles based on the names of `<creator>`, the conflicting author names result in undesirable problems in DLs. In Fig. 1, two such problems are illustrated. (a) is the screen-shot of ACM Portal where articles of a renowned computer scientist, Jeffrey D. Ullman, are "split" to ten name variants, and (b) is the screen-shot of the article list of "Dongwon Lee" in DBLP where articles of another scholar (whose name has the same spelling) are "mixed." The conflicting metadata in (a) is probably due to either errors by software systems or various names used in different articles.
   On the other hand, the conflicting metadata in (b) is due to legal mixture — when there are multiple `<creator>` (i.e., authors) with the same spelling, this problem naturally occurs. Another example is the case of "Alon Halevy," a professor at U. Washington, who used to be known as "Alon Y. Levy" before his marriage. Imagine that one performs a bibliometric analysis to measure impact factor of "Alon Halevy." Then, she would first try to gather all articles of "Alon Halevy" using `<creator>` metadata, but likely to fail to discover his articles that had the old name "Alon Y. Levy."
2. **Case of** `<publisher>` **metadata.** Another type of evolving metadata occurs when `<publisher>` entity associated with digital objects in DLs changes due to external semantic changes. Consider a scientific article with a conference/journal name as a publication venue. As time passes, often, the names of such publishers change. For instance, multiple conferences may merge into a single conference over time (e.g., both "ACM DL" and "IEEE ADL" merged into "ACM/IEEE JCDL" in 2001), or conversely a single conference can split into multiple conferences (e.g., "ACL" and "COLING" merged into "ACL-COLING" in 1998, then separated afterward). Furthermore, the characteristics of a venue may change (e.g., a workshop "ML" has evolved into a conference "ICML"). Note that if DLs have a capability to learn the evolved `<publisher>` information (i.e., "ACM DL" and "IEEE ADL" were merged to "ACM/IEEE JCDL" in 2001), then DLs may return articles from all three conferences for a user's request "find all articles in JCDL about Digital Identity after 1995" even if the query only asked about "JCDL."
3. **Case of** `<creator>`**,** `<title>`**,** `<publisher>`**, and** `<date>` **combined.** When the evolution of individual metadata is aggregated, it can result in more serious problem. Ideally, DLs should contain a single instance of a unique digital object. To achieve this, DLs often try to find all redundant instances by using their associated metadata and consolidate them. However, when metadata has conflicting and misleading values, such a de-duplication task becomes harder. Figure 2 is the screen-shot of CiteSeer and Google Scholar where users tried to locate a book "Artificial Intelligence: A Modern Approach" by Russell and Norvig. Note that both CiteSeer and Google Scholar currently keep many "redundant" instances of the same book, thinking that they are all different. However, all these instances actually refer to the identical book published by the two authors.
   The culprit of this phenomenon is that first, they do not use persistent ID system, and second, each instance of the

[1] Although the six descriptive metadata of Table 1 are only a small subset of all available descriptive metadata in Dublin Core or PREMIS data dictionary, recent study shows that majority of web and DL usage (i.e., around 30%) are the search based on author or publisher names (e.g., finding a person's home page or searching articles by some publishers). Therefore, metadata such as `<creator>`, `<publisher>`, or `<title>` has significant impact on the overall quality of digital objects and the issue of digital preservation. Furthermore, automatically detecting and fixing abnormality in administrative metadata (e.g., access control) is not only extremely difficult but also impractical. We believe that such evolving administrative (or structural for the same matter) metadata are best handled by human experts manually.

Table 1 Partial list of metadata of Dublin Core

Metadata	Definition
<creator>	An entity primarily responsible for making the content of the resource
<date>	A date associated with an event in the life cycle of the resource
<description>	An account of the content of the resource
<publisher>	An entity responsible for making the resource available
<subject>	The topic of the content of the resource
<title>	A name given to the resource

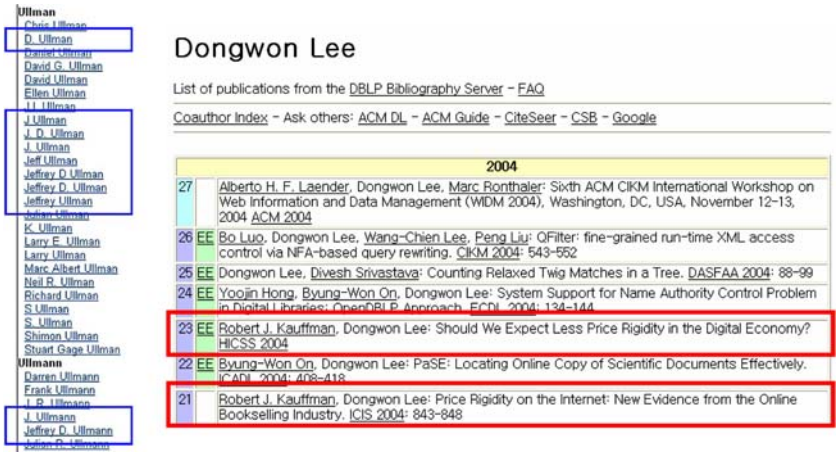


Fig. 1 Conflicting <creator> metadata in two contexts: (*left*) ten split cases of "Jeffrey D. Ullman" in ACM Portal; and (*right*) Mixed case of "Dongwon Lee" in DBLP where boxed ones are by another "Dongwon Lee"

book carries slightly different metadata (e.g., "S. Russel" vs. "Russel, Stuart" for <creator> or "Artificial Intelligence: A Modern Approach" vs. "A Modern Approach" for <title>). Since two DLs are constructed by web crawlers that automatically extract metadata from the user-posted (thus no uniform convention to follow) citation data on the Web, the quality of the extracted metadata tends to be poor, making the de-duplication task difficult, if not impossible. For DLs whose metadata are manually selected and entered by human experts (e.g., DBLP, ISI/SCI [25]), the issue of evolving metadata is less obvious, although it still exists due to data-entry errors. However, for DLs whose digital objects are systematically gathered and whose metadata are automatically generated by software agents (e.g., CiteSeer), the problem gets worse.

In addition, when different character sets are used, foreign languages with a variety of accents and special symbols are used, or a language is translated to another one, evolving (and erroneous) metadata are easily found. For instance, two <cretor> metadata — "D. Marz" and "D. März" — may confuse users of DLs if proper hints are not available.

1.2 Objectives

As demonstrated in the aforementioned three examples, the evolving and evolving metadata pauses a significant challenge to the maintenance of DLs. In particular, two critical techniques are needed to solve the evolving metadata problem: (1) *Identification* of evolving metadata in various contexts; and (2) *System Incorporation* of the identified evolving metadata.

- *Identification:* Given millions of digital objects and their associated metadata, finding the evolving metadata quickly is a daunting task. Toward this objective, what are the effective yet scalable algorithms? What are the taxonomy of all possible scenarios that evolving metadata can yield? What is the relationship of the problem to existing data integration problem (e.g., record-linkage or citation matching)?
- *System Support:* Once the evolving metadata are identified, they must be somehow fixed/updated into DLs. To support this update, what are the most basic building blocks of activities? How can they be merged or split? What are the taxonomy of all possible update scenarios? If evolving metadata are fixed, how can DLs exploit that?

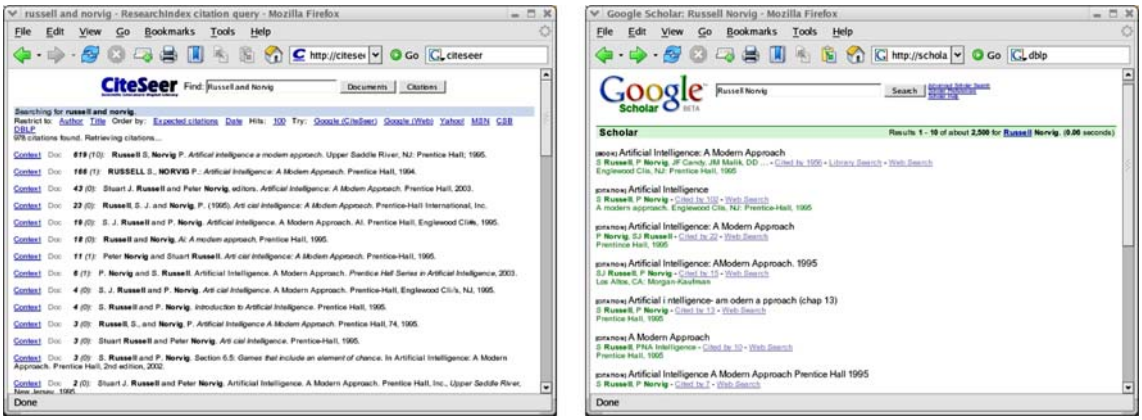


Fig. 2 Redundant instances of digital objects due to evolving metadata in CiteSeer and Google Scholar‘

Table 2 Some of the well-known scientific digital libraries

Digital library	Domain	# of digital objects (in millions)	Automatically generated?
ISI/Science Citation Index	General Sciences	25	No
CAS	Chemistry	23	No
MEDLINE/PubMed	Life Science	12	No
CiteSeer	General Sciences/Engineerings	10	Yes
eBizSearch	e-Business	N/A	Yes
arXiv e-Print	Physics, Mathematics	0.3	No
SPIRES HEP	High-energy Physics	0.5	No
DBLP	Computer Science	0.5	No
CSB	Computer Science	1.4	Yes
BibFinder	Computer Science	N/A	Yes
NetBib	Network	0.05	No

1.3 Significance

In recent years, we have witnessed a dramatic increase of DLs in both the volume of data maintained in each and the number of DLs becoming available on the Web — partly due to governmental support (e.g., NSF Digital Library Initiative), and partly due to newly available tools and standards (e.g., crawler, automatic citation indexer, OAI protocol). For a quick assessment, Table 2 summarizes the basic statistics of some of the well-known scientific DLs — either manually or automatically constructed and maintained. Note that most of them are “large-scale”, having millions of digital objects in them. One way or the other, all of these DLs have the metadata-related problem. We believe that the same problem pervades in other DLs and data applications. When the contents of DLs need to be preserved for a long period, the problem only exacerbates.

Despite the importance of the maintenance of evolving metadata in DLs, however, the conventional solution has been very limited, only keeping authority file of some sort, and never fully exploit its potential usage from the system point of view. Similarly, there has not been much study on the systematic identification of evolving metadata for large-scale DLs. The significance of this paper is to emphasize the automatic identification of those evolving metadata cases, and the role of “systems”, that is DLs, which must keep track of those changes pro-actively, and leverage on the knowledge in various services of DLs (e.g., search) to give better interactions to users. Note that the solutions that we are proposing are very practical. Therefore, they can be adopted immediately to many existing DLs, and thus can play a significant role in extending current limitations.

2 Related work

To our best knowledge, there has been little work on identifying corrupted metadata and fixing them through system support. However, a wealth of relevant works exist in slightly

different context. Here, we review two of them — name authority control and citation matching problems.

Name authority control. When popular descriptive metadata such as <creator> or <publisher> of digital objects changes in DLs, one needs to determine which of the conflicting metadata is the authoritative one. In the traditional DL community, to solve this problem, people often keep a list of authority names in a file, and when conflicting names are encountered, they consult the authority file for solution. For instance, it is known that US Library of Congress, OCLC and "Die Deutsche Bibliothek", while each maintaining large name authority databases, participate in the Virtual International Authority File project [46] to explore virtually combining the name authority files of both institutions into a single name authority service [31]. Although varies in detail, in general, a valid authority record contains a catalog heading, cross references (if any), and its justification.

Works done in [18,47] aim at detecting name variants automatically using data mining or heuristics techniques. Our proposal is more general and aims at identifying an extensive set of corrupted metadata (in addition to name variants). Also, we focus on system support issues once such variants are (semi-)automatically identified. Similarly [12] introduces a method to find matching variants of named entity in a given text such as project name (e.g., DBLP vs. Data Base and Logic Programming). The proposed specification maintains some context information and provides access to the copy of the document that an user is authorized to when multiple copies of the same document is maintained. Similarly [42] discusses an effort to standardize author names using a unique number, called INSAN. [11] is a recent implementation for name authority control, called HoPEc, which bears some similarity to our approach. The detailed comparison between our proposed approach and INSAN/HoPEc can be found in our recent work of [23].

Citation Matching. Detecting and fixing corrupted (thus conflicting) metadata is in essence similar to the problem of "database join" (thus finding similar tuples among many). In DLs context, the closest problem is known as citation matching problem (e.g., [30,38]), but it is also known as various names in various disciplines. For instance, it bears a great relevance to problems known as — record linkage (e.g., [6,14,26,48]), identity uncertainty (e.g., [38]), merge-purge (e.g., [20]), object matching (e.g., [9,44]), duplicate detection (e.g., [1,32]), approximate string join (e.g., [10, 16]) etc. In particular, by assuming records to match are stored in RDBMS, one can view record linkage as *string join* problem, but the difficulty lies in the pre-processing step to convert raw metadata string into an array of columns to fit into relational model, and some techniques [7,43] toward this problem have been proposed. There are some record linkage research work applied to specifically citation matching in DLs. For instance [27] experimented various distance-based algorithms with a conclusion that word-based matching performs well. Two-step citation matchings were proposed in [24,30] where initial rough but fast clustering (or called "Canopy") is followed by more exhaustive citation matching step. Although relevant, none of these concerns issues related to "system support" once matching or conflicting metadata are identified. Furthermore, these works do not concern the evolution of metadata and their effects on DLs.

Apart from automatic citation matching techniques, there have been a wealth of attempts to provide more fundamental solution to avoid name authority control problem. For instance, in the Open Journal project [22], hypermedia linking techniques were applied to citation linking. However, since their test-data was from SCI/ISI (relatively clean compared to ones in CiteSeer), the task of matching is arguably easier. More recently, by extending the success of the Open Journal project, the Open Citation Project (OpCit) [21,40] has developed more comprehensive solutions for the reference linking in large-scale open-access archives. Similarly [4] proposes some architectural issues and programming API for linking references. Compared to our proposal, all these are narrowly focused on the "reference" of articles with the goal of designing right models first. At the other end of the spectrum, one may assign unique and persistent identifier to each digital object. The examples include Digital Object Identifier (DOI) [3,37] or SLinkS [19]. However, the full adoption of those proposals is still far from the reality. The creation and maintenance issues of identifiers are found in [8]. Some recent work attempts to *learn* what can constitute unique identifiers [44].

Note that the evolving metadata problem that we concern in this proposal is "orthogonal" to the adoption of persistent IDs. That is, even if two instances of digital objects have the same DOI (thus one knows that they refer to the same real-world objects), both instances may still have conflicting metadata that causes problems.

3 Main proposal

In a nutshell, we consider the following question in this paper:

> When the metadata of digital objects in large-scale DLs change over time, how to quickly and accurately identify those evolving metadata in various scenarios, and fix them?

To answer this question, we explore the two inter-related tasks next. Note that our goal in this paper is *not* to propose a particular algorithm. Rather, we advocate the importance of

the problem and the framework of algorithmic solution and system support.

3.1 Identifying evolving metadata

When metadata is corrupted in a large-scale DL, it is challenging to automatically detect the corrupted metadata in an effective and scalable way. To make matters worse, depending on the type of digital objects and their metadata, the identification algorithm is likely to vary. Below, let us present two specific algorithms to identify corrupted *descriptive* metadata in two different contexts. To facilitate the presentation, let us first assume that the digital object, o, is the "citation" of an article, and the corrupted metadata, m, is `<creator>`, the author list of the article.

1. The Mixed Object (MO) Problem and Labeling Algorithm. In Fig. 1b, due to the corrupted `<creator>` metadata, digital objects (i.e., scientific articles) by different authors are mixed — we refer to this as *Mixed Object* (*MO*) problem. Consider a collection of digital objects, O, all with `<creator>` metadata set to an author a_i. Now, suppose some of the digital objects in O have corrupted `<creator>` metadata and should have been labeled to another author a_j. How can we identify (1) those objects with corrupted metadata? and (2) the correct value of the corrupted metadata? The challenge here is that since two different authors, a_i and a_j, have the "same" name spellings in the `<creator>` metadata (thus they are corrupted), one cannot easily differentiate one from corrupted metadata by simply using distance between their names (e.g., Edit distance).

One way to overcome this is to exploit additional metadata information. Suppose a DL has a set of citations C by an author a_i, and needs to identify citations $C'(\subseteq C)$ that are "not" by a_i. Note that `<creator>` metadata of C' still contains a value of a_i. Here, we may use additional metadata information such as coauthor list, common keywords used in the titles of articles, or common publication outlets, etc. Consider a citation c_i with a set of coauthors $A = \{a_1, \ldots, a_n\}$, a set of keywords from the title $T = \{t_1, \ldots, t_m\}$, and a venue name V. Then, after removing the i-th coauthor a_i $(\in A)$, we try to "guess" back the removed author using associated information. Let us call this procedure of guessing as the *Labeling* algorithm. If we assume that there is a "good" labeling function $f_{cl} : c_i \rightarrow a_j$, then, the above MO problem can be casted as follows. Given citations C by an author a_1:

for each citation c_i $(\in C)$:
  remove a_1 from coauthor list of c_i;
  f_{cl} is applied to get a_2 (i.e., guessed name);
  if $a_1 \neq a_2$, then c_i has corrupted metadata;
  remove c_i from C;

At the end, C is left with only citations with correct metadata (i.e., those written by a_1) and those removed from C are ones with corrupted metadata (e.g., citations written by another author a_2 with the same name spelling). Therefore, if one can find a good labeling function f_{cl}, then one can solve the MO problem. Suppose one wants to "label" a collection of citations, C, against a set of possible authors A. A naive algorithm, then, is (let ϕ be a similarity measure[2] between a citation c_i and an author a_j):

for each citation c_i $(\in C)$:
  examine all names $a_j (\in A)$;
  return a_j $(\in A)$ with MAX ϕ;

This baseline approach presents two technical challenges: (1) Since c_i and a_j are two different entities to compare in real world (i.e., c_i is a digital object and a_i is one of its metadata), the choice of good similarity measure is critical; and (2) When a DL has a large number of citations and authors in the collection, the baseline approach with a time complexity, $O(|C||A|)$,[3] is prohibitively expensive to run (e.g., the DBLP has about 0.56 million authors). Toward these two issues, we propose the following.

First, to measure the similarity between object and its metadata, one can use the idea similar to that in [18,28,35,36], representing a citation as 3-tuple of coauthors, titles, and venues — then, the similarity between a citation c and an author a (hereafter, $sim(c, a)$) can be estimated as the similarity between a 3-tuple representation of c and that of a:

$$sim(c, a) = \alpha\ sim(\mathbf{c_c}, \mathbf{a_c}) + \beta\ sim(\mathbf{c_t}, \mathbf{a_t}) + \gamma\ sim(\mathbf{c_v}, \mathbf{a_v})$$

where $\alpha + \beta + \gamma = 1$ (i.e., weighting factors), $\mathbf{c_c}$, $\mathbf{c_t}$, and $\mathbf{c_v}$ are token vectors of coauthors, paper titles, and venues, respectively, of the citation c, and $\mathbf{a_c}$, $\mathbf{a_t}$, and $\mathbf{a_v}$ are token vectors of coauthors, paper titles, and venues from "all" citations of the author a, respectively. The intuition is that if a metadata is corrupted, then it is likely to contain an abnormal value that is vastly different from the "collective" values of the normal metadata. In turn, each similarity measure between two token vectors can be estimated using the standard IR techniques such as the *cosine similarity*, $\cos(\theta) = \frac{\mathbf{v} \cdot \mathbf{w}}{\|\mathbf{v}\| \cdot \|\mathbf{w}\|}$, along with TF/IDF.[4]

[2] In this paper, we use both terms of distance and similarity somewhat interchangeably. We assume that there is a straightforward conversion method between "distance" and "similarity" of two objects (e.g., distance = 1 − similarity).

[3] The Big O notation, $O()$, is a mathematical tool to describe the asymptotic upper bound of functions for a large input set. For instance, $O(|C||A|)$ sets the upper bound (i.e., worst case) of the algorithm.

[4] The TF/IDF (term frequency/inverse document frequency) is a popular weighting scheme in Information Retrieval, where the importance of a token increases proportionally to the frequency of a token in the document but decreases by the frequency of the token in the entire corpus.

Example 1 For instance, a citation c "E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Commun. ACM 13(6): 377–387 (1970)" is represented as: $\mathbf{c_c}$ = ["E.F. Codd"], $\mathbf{c_t}$ = ["Relational", "Model", "Data", "Large", "Shared", "Data", "Banks"], and $\mathbf{c_v}$ = ["Commun.", "ACM"]).[5] Similarly, an author "John Doe" with two citations ("John Doe, John Smith: Data Quality Algorithm, IQIS, 2005", and "Dongwon Lee, John Doe, Jaewoo Kang: Data Cleaning for XML, ACM Joint C. on Digital Libraries, 2005") is represented as: $\mathbf{a_c}$=["John Doe", "John Smith", "Dongwon Lee", "Jaewoo Kang"], $\mathbf{a_t}$=["Data", "Quality", "Algorithm", "Cleaning", "XML"], and $\mathbf{a_v}$=["IQIS", "ACM", "Joint", "C.", "Digital", "Libraries"]. Then, the similarity of the citation c and an author "John Doe" is equivalent to: $sim(c, a)$. That is, if $sim(c, a)$ is beyond some threshold, we "guess" that c is a false citation and should have been labeled under "John Doe", not "E. F. Codd" (false positive case). When there are many such authors, we label c as the author with the maximum $sim(c, a)$.

Second, to be scalable, one may adopt an approximation algorithm in measuring the similarity. Note that for a citation c, one does not need to check if c can be labeled as an author a for the entire authors of A. If one can quickly determine candidate author set from all authors (i.e., pre-filtering), then c is better off to be tested against only the authors in candidate set. For the pre-filtering, we may use the sampling technique developed for approximate join algorithm [16]. Let $S(\subseteq A)$ be the sample candidate set that is drawn probabilistically. Then:

for each citation c_i $(\in C)$:
  draw a sample $S(\subseteq A)$;
  examine all names $s_j(\in S)$;
  return s_j $(\in S)$ with MAX ϕ;

Note that the complexity is reduced to $O(|A| + |C||S|)$, that is typically more scalable than $O(|C||A|)$ since $|S| \ll |A|$.

2. The Split Object (SO) Problem and Disambiguation Algorithm. In Fig. 1a, due to the corrupted <creator> metadata, digital objects (i.e., scientific articles) by the same author are split — we refer to this as *Split Object (SO)* problem. In the citation vs. author context, the SO problem can be casted as follows: Given two lists of author names, X and Y, for each author name x $(\in X)$, find name variants of $x : y_1, y_2, \ldots, y_n$ $(\in Y)$. That is, all of $y_1 \ldots y_n$ are corrupted metadata, and should have been set to x instead. The baseline approach to solve the problem is to treat each value of metadata as a "string", and compute all pair-wise string similarity using some function, $dist(x, y)$:

for each metadata x $(\in X)$:
  for each metadata y $(\in Y)$
    if $sim(x, y) > \phi$, x and y are variants;

Note that if x and y are variants, then one is the corrupted form of the other (canonical one). This baseline approach has the limitations similar to the baseline approach of the MO problem. Since the baseline approach is prohibitively expensive to run for large DLs (because of its quadratic time complexity, $O(|X||Y|)$)), there is a need for more *scalable* algorithms. Furthermore, some metadata from similar cultural or national background shares similar spellings (e.g., <creator>), and thus those algorithms should not be too much dependent on the syntactic similarities of metadata. For instance, given 10 articles with <creator> metadata = "Dongwon Lee", and another 20 articles with <creator> metadata = "D. Lee", to determine if one metadata is the corrupted of the other, one may test if there is any correlation between the associated metadata (e.g., coauthor lists) of "Dongwon Lee" and "D. Lee." By using the same sampling idea, this *Disambiguation* algorithm to identify split digital objects can be written as follows (x, y, and z are metadata):

/* let C_a be coauthor information of author a; */
for each x $(\in X)$, draw a sample set $S_x(\in S)$;
for each y $(\in Y)$: . . . . . /* Step 1 */
  assign y to all relevant samples $S_i(\in S)$;
for each sample S_x $(\in S)$: . . . . . /* Step 2 */
  for each z $(\in S_x)$:
    if $sim(C_x, C_z) > \phi$:
      x and z are corrupted metadata;

Note that the time complexity after sampling becomes $O(|X| + |Y| + \mathcal{C}|S|)$, where $\mathcal{C}$ is the average number of metadata per sample. In general, $\mathcal{C}|S| \ll |X||Y|$, making the algorithm more scalable.

Depending on the choices in both Steps 1 and 2, many variations of the disambiguation algorithm are feasible. Let us use the following notations: (1) 1-N is a single-step pair-wise name matching scheme without using sampling or coauthor information (i.e., it uses plain pair-wise author name comparison); (2) 2-NN uses the two-step approach, but do not exploit "coauthor" information; (3) 2-NC is a two-step approach using the sampling and exploiting "coauthor" information instead of author names; and (4) 2-NH is the modification of 2-NC in that in step 2, it combines both author and coauthor information together with proper weights (e.g., In experimentations, we used one-fourth and three-fourth for author and coauthor, respectively). Although using the sampling speeds up the whole processing significantly, another important issues is to find out the right distance metric to use in Step 2. Since each metadata (e.g., author) is represented as a potentially very long list of associated metadata (e.g., coauthor list), different distance metrics tend to show different accuracy/performance trade-off. Our preliminary results for all these variations are to follow in Sect. 4.

[5] We assume that all stop-words are pre-pruned from the title.

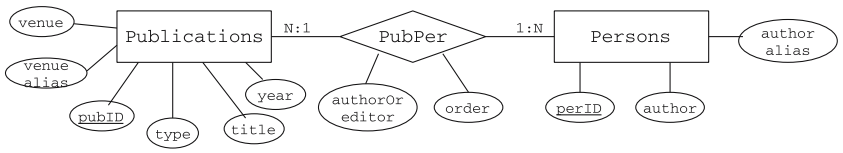


Fig. 3 One possible ER diagram for simple bibliographic digital libraries

What we have presented so far is two common (mixed and split object) problems and corresponding algorithmic framework. Specific implementation under the proposed framework will vary. In Sect. 4, for instance, a few choices of implementations are explored for validating the proposed ideas.

3.2 Fixing and using evolving metadata

Once evolving metadata is identified (manually by a librarian/author or automatically by the algorithms of Sect. 3.1), the findings must be updated into DLs to solve the evolving metadata. When two metadata are evolving and conflicting each other, three simple patterns are plausible:[6]

1. **Linear change** ($A \rightarrow B$). A metadata A is changed to B. For instance, a `<creator>` or `<publisher>` metadata is changed over the years.
2. **Split** ($A \rightarrow \{A_1, A_2\}$). A value of metadata is split into multiple ones. For instance, a conference C can be broken into two C_1 and C_2. Then, the metadata of C appears to be broken to that of C_1 and C_2.
3. **Merge** ($\{A_1, A_2\} \rightarrow A$). Conversely, multiple metadata can be merged into one. For instance, two variants of a person's name may be merged into one authoritative one.

In order to demonstrate our idea, let us imagine a bibliographic DL built on RDBMS with a simple[7] ER diagram shown in Fig. 3: (1) `Publications` table contains citations of each publication, except their author names, (2) `Persons` table contains author-related information, and (3) `PubPer(pubID, perID, ...)` tells which publication is authored by which person (by two foreign keys). Since a publication can be authored by many co-authors together, to avoid 1NF violation, separate `Persons` table is needed. Also, note that there are placeholders to store the "alias" name variants, for both author name (`authorAlias`) and publication venue (`venueAlias`). Now, using this simple structure, let us discuss how three patterns of evolving metadata can be "fixed". Here, two metadata, `<creator>` and `<publisher>`, are captured as `author` and `venue`. The ER diagram shown in Fig. 4 is another possible one in FRBR model [45], and conceptually equivalent to that in Fig. 3. Subsequent discussion is based on the ER design in Fig. 3.

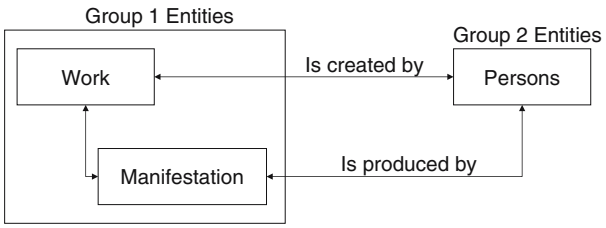


Fig. 4 Another possible ER diagram in FRBR [45] that is equivalent to that in Fig. 3

1. *Linear change.* When a metadata A is changed to B, the system sets A as an alias (either in the `authorAlias` or `venueAlias` column), and sets B as the current name (either in the author or venue column). Further, A's publications are moved to B by changing (`pubID`,`perID`) pair in `PurPer` table.
2. *Split.* Suppose a metadata A needs to be split into B and C. Then, the system needs to know not only new names of A, (i.e., B and C), but also which of the A's publications belong to either B or C. Then, the system creates a new unique ID, `perID`, for both B and C and moves their corresponding publications accordingly. Note that B and C can be the same name. For instance, when the publications of two "Wei Wang"s were incorrectly mixed, separating them out is the case of split as in "Wei Wang" $\rightarrow$ {"Wei Wang$_1$", "Wei Wang$_2$"}.
3. *Merge.* When two metadata, A and B, are conflicting and need to be merged into C, the system sets both A and B as aliases of C. Since all of A, B, C still have unique ID, `perID`, in the system, if a user wants, she can still search using old metadata A and B.

By combining the three core elements as basic building blocks, one can cover various patterns of evolving metadata found. Some of our preliminary taxonomy can be found in [23].

[6] More variety of evolving metadata patterns for book-like publications are possible. For instance, Functional Requirements for Bibliographic Records (FRBR) [45] provides a conceptual model and means to specify various patterns such as 3rd edition of a book or German translation of an English book "X". Since the focus of this article is to demonstrate system support to handle such evolving metadata pattern, however, we only focus on the rather simple three patterns.

[7] Note that this ER diagram is too simple to handle the preservation of various metadata in real applications. We simply use the current form to facilitate the presentation.

Once evolving metadata is identified and updated into the system, those knowledge can be exploited further in various functionalities of DLs. Suppose a user is searching for all publications about XML by "Alon Halevy." When she submits two keywords "Alon Halevy" in the search box of DLs, internally, an SQL query similar to the following (assuming the physical schema of underlying database is created based on Fig. 3) will be issued:

```
SELECT P1.*
FROM   Publications P1, PubPer P2, Persons P3
WHERE  P1.pubID=P2.pubID AND P2.perID=P3.perID AND
       P3.author = 'Alon Halevy' AND
       P1.title LIKE '%XML%'
```

However, what this user did not know is that DL keeps a separate list of publications by the same physical person, but under different name "Alon Levy" (i.e., different <creator> metadata). When such related information was updated by the aforementioned steps, however, the DL can exploit the knowledge — e.g., alert warning message to users, return merged list, or display a link to publication lists under related name variants, etc. For instance, the following SQL query would return a merged list using "alias" columns:

```
(SELECT P1.*
   FROM   Publications P1, PubPer P2, Persons P3
   WHERE  P1.pubID=P2.pubID AND P2.perID=P3.perID
          AND P3.author = 'Alon Halevy'
          AND P1.title LIKE '%XML%')
UNION
  (SELECT P1.*
   FROM   Publications P1, PubPer P2, Persons P3,
          Persons P4
   WHERE  P1.pubID=P2.pubID AND P2.perID=P4.perID
          AND P3.author = 'Alon Levy' AND
          P1.title LIKE '%XML%' AND
          P3.authorAlias = P4.author)
```

That is, when corrupted metadata are identified and corrected by the system, in this scenario, that knowledge was captured in the alias column. Therefore, as shown, by simple equi-join of SQL, the DL can support such a tracking easily.

According to three patterns of linear, split, and merge, over the time dimension, there are various strategies as to how DLs can react to such a search function. Suppose conferences (i.e., <publisher> metadata), c_1 to c_8, have evolved as follows: (1) $c_1 \rightarrow c_2$ (i.e., linear name change); (2) $c_3 \rightarrow c_4, c_5$ (i.e., conference split); and (3) $c_6, c_7 \rightarrow c_8$ (i.e., conference merge). Then, three possible search strategies that can exploit the fact that corrupted metadata was fixed are illustrated in Table 3.

Both "backward" and "forward" schemes are temporal strategies where the system searches related conferences toward backward or forward on a temporal dimension. For instance, in the backward strategy, when a user searches for c_2, system shows c_2 as well as all its predecessors, c_1, as answers (i.e., all digital objects with conflicting metadata in past). Another possible strategy is the "semantic" search, where all semantically related results are returned, regardless of the temporal aspect (i.e., show all digital objects with conflicting metadata). Note that the semantic strategy is equivalent to the union of both backward and forward strategies. For instance, since c_3 is broken into c_4 and c_5, whenever browsing c_3 occurs, it is expanded to all conferences related to c_3, thus c_4 and c_5. Note, however, that browsing c_4 is not expanded to c_5.

In addition to using temporal aspect, other search strategies can be devised. Similarly, other functionalities of DLs (e.g., browsing, publish/subscribe, or registration) may adopt to exploit the new knowledge.

Table 3 Various search strategies. Q and A refer to "query" and "answer," respectively

Backward		Forward		Semantic	
Q	A	Q	A	Q	A
c_1	c_1	c_1	c_1, c_2	c_1	c_1, c_2
c_2	c_1, c_2	c_2	c_2	c_2	c_1, c_2
c_3	c_3	c_3	c_3, c_4, c_5	c_3	c_3, c_4, c_5
c_4	c_3, c_4	c_4	c_4	c_4	c_3, c_4
c_5	c_3, c_5	c_5	c_5	c_5	c_3, c_5
c_6	c_6	c_6	c_6, c_8	c_6	c_6, c_8
c_7	c_7	c_7	c_7, c_8	c_7	c_7, c_8
c_8	c_6, c_7, c_8	c_8	c_8	c_8	c_6, c_7, c_8

4 Experimental result and prototype

The aforementioned labeling and disambiguation algorithms of Sect. 3.1 have been implemented using bibliographic DLs, and various configurations were examined. The target metadata that we tried to identify was <creator>, and the digital object was citations of the four DLs.

Set-up. We have gathered real citation data from four different domains, as summarized in Table 4. These are substantially "large-scale" DLs (e.g., DBLP has 360 K authors and 560 K citations in it). Different disciplines appear to have slightly different citation policies and conventions. For instance, Physics and Medical communities seem to have more number of coauthors per article than Economics community. Furthermore, the conventions of citation also vary. For instance, citations in e-Print use the first name of authors as only initial, while ones in DBLP use full names. All four data sets are pre-segmented (i.e., each field of coauthors, title, and venue are already known to us). For the sampling technique, we used the implementation of [17] with a

Table 4 Summary of data sets

Data set	Domain	# of authors/ # of citations	# of coauthors per author (avg/med/std-dev)	# of tokens in coauthors per author (avg/med/std-dev)
DBLP	CompSci	364,377/562,978	4.9/2/7.8	11.5/6/18
e-Print	Physics	94,172/156,627	12.9/4/33.9	33.4/12/98.3
BioMed	Medical	24,098/6,169	6.1/4/4.8	13.7/12/11.0
EconPapers	Economics	18,399/20,486	1.5/1/1.6	3.7/3/4.1

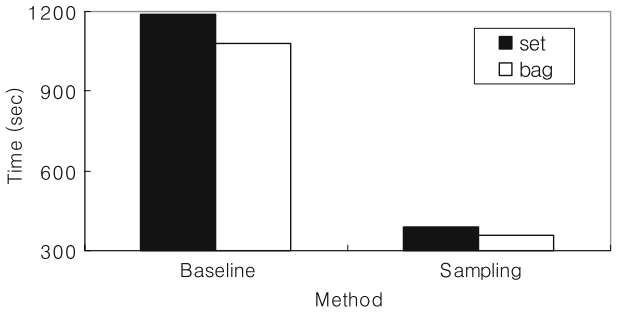


Fig. 5 Scalability (EconPapers)

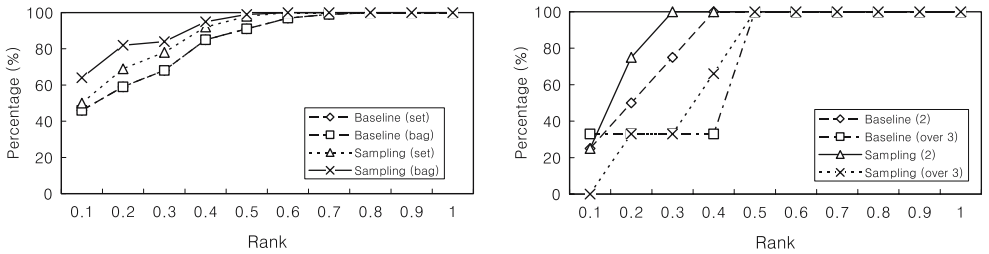
sample $S = 64$ and a threshold $\theta = 0.1$. For the distance metrics, we have considered two supervised methods (i.e., Naive Bayes Model and Support Vector Machine) and five unsupervised methods (i.e., cosine, TF/IDF, Jaccard, Jaro, and Jaro Winkler). For supervised learning methods, citations per author are randomly split, with half of them used for training, and the other half for testing. For the implementation of Support Vector Machines, LIBSVM [15] was used. For the string-based distance functions of the unsupervised learning methods, we used the implementations of TF/IDF, Jaccard, Jaro, and JaroWinkler from SecondString [41]. Other remaining methods were implemented by us in Java. All experimentation was done using Microsoft SQL Server 2000 on Pentium III 3 GHZ/512MB.

Results of the MO Problem. For the MO problem, we used two DLs out of four as test-beds: DBLP and EconPapers. For DBLP (which the author is familiar with), we collected real examples exhibiting the MO problem: e.g., "Dongwon Lee", "Chen Li", "Wei Liu", "Prasenjit Mitra", and "Wei Wang", etc, and for EconPapers (which the author is not familiar with), we injected artificial "false citations" into each author's citation collection (thus creating corrupted metadata). For both data sets, we tested how to find the "false citations" from an author's citations (that is, we had a solution set for both cases). In constructing token vectors, we used two models, *Set* and *Bag*, depending on the preservation of the multiple occurrences of the same token. For testing, we used the weights, $\alpha = 0.5$, $\beta = 0.3$, and $\gamma = 0.2$. As evaluation metrics, we used *time* for scalability, and *percentage/rank* ratio for accuracy (i.e., A false citation c_f must be ranked low in $sim(c_f, a)$. Thus, we measured how much percentage of false citations were ranked in the bottom 10, 20%, etc).

First, Fig. 5 clearly shows the superior scalability of the sampling-based approach over the baseline one (upto 4 times faster), regardless of set or bag models. Since the time complexity of the sampling-based approach is bounded by S, which was set to 64, for a large C such as DBLP, the scalability gap between two approaches further widens. Second, Fig. 6(a) illustrates the accuracy of both approaches for EconPapers. For instance, when there is a single false citation c_f hidden in the 100 citations, the sampling approach with the bag model can identify c_f with over 60% accuracy (i.e., rank = 0.1/% = 64). Furthermore, when it can return upto 2 citations as answers, its accuracy improves to over 80% (i.e., rank = 0.2/% = 82). Since many tokens in citations tend to co-occur (e.g., same authors tend to use the same keywords in titles), the bag model that preserves this property performs better.

Finally, Fig. 6(b) shows results on DBLP using only the bag model (the set model showed similar pattern as well and is omitted). Note that some collection has a mixture of "2" authors' citations while others have that of "over 3" authors (e.g., there exists more than 3 authors with the same spellings of "Wei Liu"—three-way corruption of `<creator>` metadata). Intuitively, collections with more number of authors' citations mixed are more difficult to handle. For instance, when 2 authors' citations are mixed, 100% of false citations are always ranked in the lower 30% (i.e., rank = 0.3) using the sampling approach. However, when more than 3 authors' citations are mixed, the percentages drop to mere 35% — it is very difficult to decipher a false citation when it is hidden in a collection that contains a variety of citations from many authors.

Results of the SO Problem. To make a solution set, for each data set, we prepare two lists of author names, X and Y, where X contains randomly chosen 100 names, and Y contains the entire author names (e.g., 364,377 names for DBLP). For instance, for "Grzegorz Rozenberg" with 344 citations and 114 coauthors in DBLP, we create a new name like "G. Rozenberg" (abbreviation of the first name) or "Grzegorz Rozenbergg" (typo in the last name). Then, after



(a) Econ Papers **(b)** DBLP

Fig. 6 Accuracy of the labeling algorithm

splitting the original 344 citations into halves, each name carries half of citations, 172, and is put back into X and Y, respectively. Then, through the proposed two-step disambiguation algorithm, for each name in X, we test if the algorithm is able to find the corresponding artificial name variant in Y (that we generated and thus know what they are). Note that the way we generate artificial name variants may affect the performance of the algorithm. In general, it is difficult to precisely capture the right percentages of different error types in author name variants. For the original name "Ji-Woo K. Li", for instance, some of possible error types are name abbreviation ("J. K. Li"), name alternation ("Li, Ji-Woo K."), typo ("Ji-Woo K. Lee" or "Jee-Woo K. Lee"), contraction ("Jiwoo K. Li"), omission ("Ji-Woo Li"), or combinations of these. To quantify the effect of error types on the accuracy of the disambiguation algorithm, we first compared two cases: (1) mixed error types of abbreviation (30%), alternation (30%), typo (12% each in first/last name), contraction (2%), omission (4%), and combination (10%); and (2) abbreviation of the first name (85%) and typo (15%). At the end, regardless of the error types or their percentages, both cases showed reasonably similar accuracies for all seven distance metrics (i.e., 0.8–0.9 accuracy). Here, we only show the latter case (85%/15%).

Figure 7 summarizes the experimental results of four alternatives using three representative metrics — TF/IDF, Jaccard, and Jaro. In terms of the processing time, 1-N is the slowest for TF/IDF and Jaccard, as expected, due to its quadratic time complexity (i.e., $100 \times 364,377$ times of pair-wise name comparisons). The other three show similar performance thanks to the sampling. In terms of accuracy, both 2-NC and 2-NH shows about 20–30% improvement, compared to 1-N and 2-NN, validating the assumption — exploiting additional information (i.e., coauthor) than the simple name spelling is more beneficial. Since 2-NH shows no noticeable improvements over 2-NC, in the remaining experiments, we use 2-NC as a default scheme.

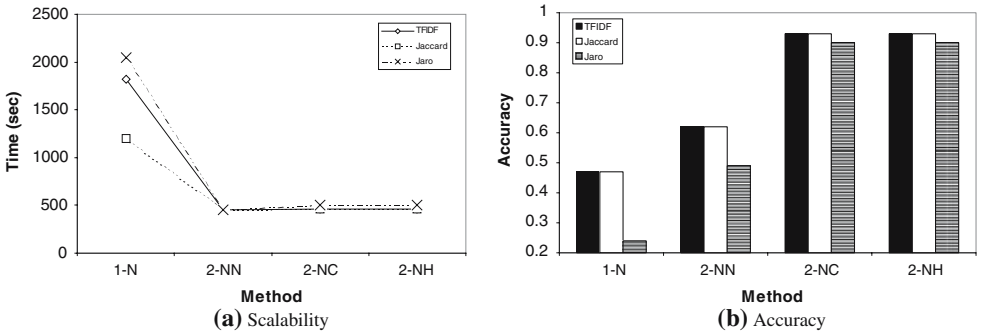
Next, we measured the processing time for step 2 alone (distance measure stage) as shown in Fig. 8a. In general, token-based distance metrics (e.g., TF/IDF, Jaccard) outperforms edit distance based metrics (e.g., Jaro, JaroWinkler). This becomes clearly noticeable for DBLP, but not for EconPapers for its small size. In addition, SVM tends to take more time than the others since the hyperplane needs to be split in succession due to SVM's binary-classifiers. Figure 8b summarizes the accuracies of our proposal for all four data sets (with $k = 5$). In general, the distance metrics such as the SVM, cosine, TF/IDF and Jaccard perform much better than the others. For DBLP data set, most distance metrics achieved upto 0.93 accuracy, finding most of 100 name variants out of 364,377 candidates. For e-Print data set, the accuracy drops down, except the SVM, and for BioMed data set, it gets worse (especially for Jaro and JaroWinkler).

The accuracies of DBLP and e-Print data sets are better than that of BioMed data set. The poor performance of BioMed case is mainly due to the small number of citations per authors in data set. Since 2-NC scheme is exploiting coauthor information of the author in question to find name variants, the existence of "common" coauthor names is a must. However, in the BioMed data set, each author has only a small number of citations, 1.18, on average, and only small number of coauthors, 6.1, on average, making a total number of coauthors as $7.19 = 1.18 \times 6.1$ (assuming all coauthors are distinct). Therefore, for two arbitrary author names x and y, the probability of having "common" coauthors in BioMed data set is not high. On the other hand, for the e-Print data set, the average number of citations (resp. coauthors) per author is higher, 4.27 (resp. 12.94), making a total number of coauthors as $55.25 = 4.27 \times 12.94$ — roughly 8 times of the BioMed data set.

The preliminary results validate that our initial ideas are feasible to detect evolving metadata in large-scale DLs, but there are ample rooms for improvement.

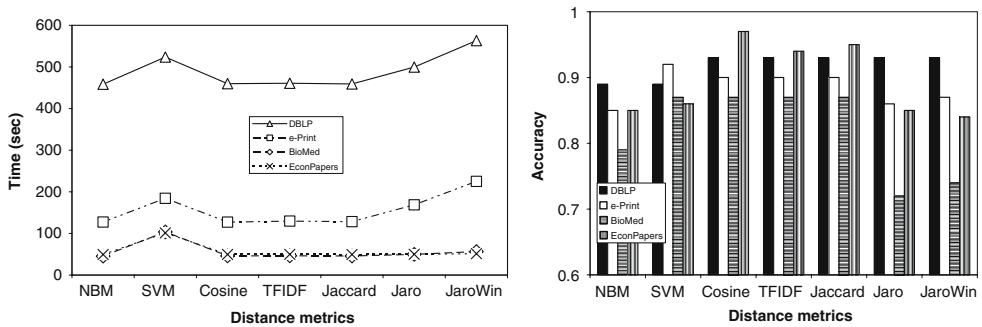
Prototype for System Support. Authors have recently built a prototype system, called OpenDBLP[8] [23]. OpenDBLP is a novel DL with the idea of providing advanced database features and web services interfaces to conventional (static and agent unfriendly) DL. OpenDBLP is (1) relatively smaller-scaled (i.e., having only about 500,000 citations without

[8] http://opendblp.psu.edu/.



(a) Scalability **(b)** Accuracy

Fig. 7 Scalability and Accuracy (DBLP when top-1 is considered)



(a) Processing time for Step 2. **(b)** Accuracy (when top-5 are considered)

Fig. 8 Comparison of four data sets

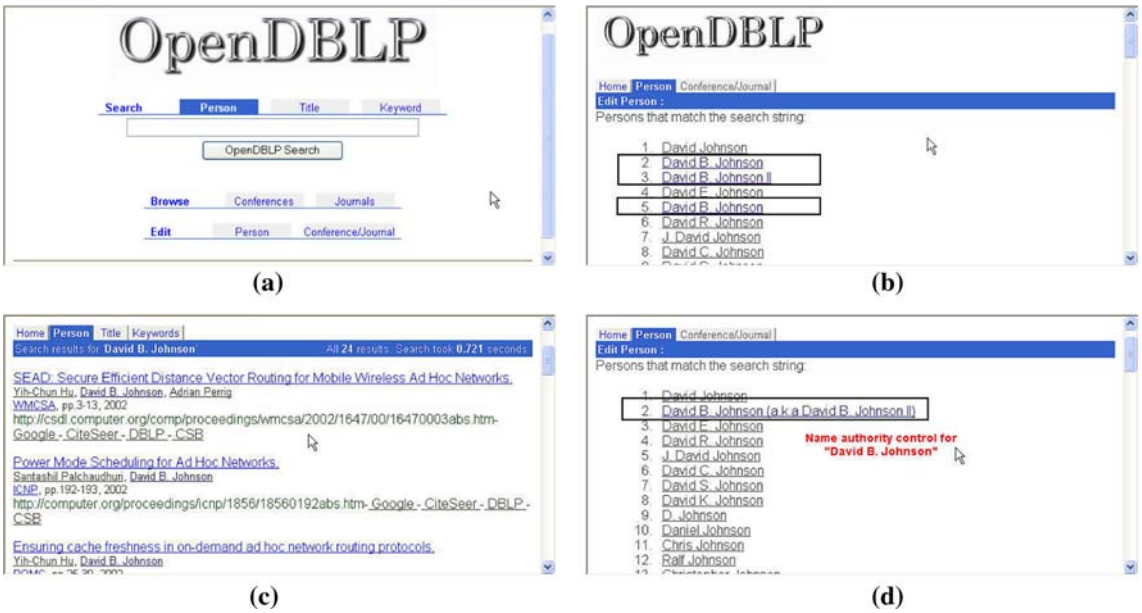


Fig. 9 The OpenDBLP implementation: **a** OpenDBLP main, **b** Search result of "David Johnson", **c** Publications of "David B. Johnson" among many "Johnson's", and **d** Result after "merging" several "David B. Johnson"

having actual articles stored); (2) on Computer Science domain only (i.e., the original data came from the well-known DBLP in computer science); and (3) open source database based (i.e., MySQL v 4.0.5). OpenDBLP was used as a test-bed to experiment the evolving metadata idea. Therefore, in OpenDBLP, we have fully implemented the algorithms and system support proposed in Sects. 3.1 and 3.2.

Suppose a user wants to retrieve publications of "David Johnson". In Fig. 9a, she types "david johnson" and gets all authors that partially match the name (Fig. 9b). Then, she drills down to "David B. Johnson" by clicking the link to it (Fig. 9c). To illustrate the evolving metadata issue, let us suppose that the user notices that some publications of the "David B. Johnson" are mixed with name variant

"David B. Johnson II", etc., and report the bug to the administrator. By using the "merge" pattern of Sect. 3.2, then, the administrator fixes the problem. After the update, the same author's publications are "automatically" consolidated by the OpenDBLP and all known name variants (i.e., alias) are displayed together to help users as shown in Fig. 9d. Therefore, Fig. 9a-d illustrates how to *update* and *search* evolving metadata in DLs by the help of the "systems".

5 Conclusion

We have proposed practical system-oriented solutions toward maintenance of evolving metadata that often occurs in long-term digital libraries and archives. We have presented scalable algorithmic framework to identify evolving metadata quickly, and argue that system support to handle evolving metadata is much needed. By showing promising results and prototype system, we demonstrate the feasibility of our proposal.

However, needless to say, many research directions are ahead. For instance, more scalable and versatile yet accurate algorithms to identify evolving metadata are needed. In this paper, we have focused on only a few descriptive metadata that are commonly found in publication DLs. More research is needed to cover more diverse domains and metadata types. In addition, more extensive study as to the relationship and taxonomy between evolving metadata and systems is needed. The proposed simple three types of linear change, split, and merge may not be sufficient for more complex applications. In particular, it would be interesting to investigate how to support all those richer patterns introduced in FRBR model using the system-oriented approach that we are advocating in this article.

Acknowledgments We thank Yoojin Hong and Byung-Won On for their contribution to the OpenDBLP and Quagga[9] projects which this manuscript is partially based on. This research has been partially supported by IBM Eclipse Innovation Award (2004, 2006) and Microsoft SciData Award (2005).

References

1. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: VLDB (2002)
2. arXiv.org e Print archive. http://arxiv.org/
3. Atkins, H., Lyons, C., Ratner, H., Risher, C., Shillum, C., Sidman, D., Stevens, A., Arms, W.: Reference linking with DOIs: a case study. D-Lib Magazine (2000)
4. Bergmark, D., Lagoze, C.: An architecture for automatic reference linking. In: European Conf. on Digital Libraries (ECDL), Darmstadt, Germany (2001)
5. Digital Bibliography and Library Project (DBLP). http://dblp.uni-trier.de/
6. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name-matching in information integration. IEEE Intell. Syst. **18**(5), 16–23 (2003)
7. Borkar, V.R., Deshmukh, K., Sarawagi, S.: Automatic segmentation of text into structured records. In: ACM SIGMOD, Santa Barbara (2001)
8. Caplan, P., Arms, W.: Reference linking for journal articles. D-Lib Magaz., **5**(7/8) (1999) http://www.dlib.org/dlib/july99/caplan/07caplan.html
9. Chaudhuri, S., Ganjam, K., Ganti, V., Motwani, R.: Robust and Efficient Fuzzy Match for Online Data Cleaning. In: ACM SIGMOD (2003)
10. Cohen, W.W.: Data integration using similarity joins and a word-based information representation language. ACM Trans. Inf. Syst. (TOIS) **18**(3), 288–321 (2000)
11. Cruz, J.M.B., Klink, N.J.R., Krichel, T.: Personal data in a large digital library. In: European Conf. on Digital Libraries (ECDL) (2000)
12. Davis, P.T., Elson, D.K., Klavans, J.L.: Methods for precise named entity matching in digital collection. In: ACM/IEEE Joint Conf. on Digital Libraries (JCDL) (2003)
13. DCMI. Dublin Core Metadata Initiative. Web page. http://dublincore.org/.
14. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. J. Am. Stati. Soc. **64**, 1183–1210 (1969)
15. A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
16. Gravano, L., Ipeirotis, P.G., Koudas, N., Srivastava, D.: Text joins for data cleansing and integration in an RDBMS. In: IEEE ICDE, (2003)
17. Gravano, L., Ipeirotis, P.G., Koudas, N., Srivastava, D.: Text joins in an RDBMS for web data integration. In: Int'l World Wide Web Conf. (WWW) (2003)
18. Han, H., Giles, C.L., Zha, H., Li, C., Tsioutsiouliklis, K.: Two supervised learning approaches for name disambiguation in author citations. In: ACM/IEEE Joint Conf. on Digital Libraries (JCDL), Jun. (2004)
19. Hellman, E.: Scholarly Link Specification Framework (SLinkS), Nov. 1998. http://www.openly.com/SLinkS/
20. Hernandez, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: ACM SIGMOD (1995)
21. Hitchcock, S., Brody, T., Gutteridge, C., Carr, L., Hall, W., Harnad, S., Bergmark, D., Lagoze, C.: Open Citation Linking: The Way Forward. D-Lib Magaz. 8(10) (2002)
22. Hitchcock, S., Carr, L., Hall, W., Harris, S., Probets, S., Evans, D., Brailsford, D.: Linking electronic journals: lessons from the open journal project. D-Lib Magaz (1998)
23. Hong, Y., On, B.-W., Lee, D.: System support for name authority control problem in digital libraries: OpenDBLP approach. In: European Conf. on Digital Libraries (ECDL), Bath (2004)
24. Hylton, J.A.: Identifying and Merging Related Bibliographic Records. PhD thesis, Dept. of EECS, MIT, LCS (1996) Technical Report MIT/LCS/TR-678
25. ISI/Science Citation Index. http://www.isinet.com/
26. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. J. Ame. Stat. Assoc, **84**(406) (1989)
27. Lawrence, S., Giles, C.L., Bollacker, K.: Digital Libraries and autonomous citation indexing. IEEE Comput. **32**(6), 67–71 (1999)
28. Lee, D., On, B.-W., Kang, J., Park, S.: Effective and scalable solutions for mixed and split citation problems in digital libraries. In: ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS), Jun. (2005)

[9] http://pike.psu.edu/quagga/.

29. CiteSeer: Scientific Literature Digital Library. http://citeseer.ist.psu.edu/
30. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. In: ACM KDD, Boston (2000)
31. Miner, R.: Enhancing the Searching of Mathematics, Jun. (2004) http://www.ima.umn.edu/complex/spring/math-searching.html
32. Monge, A.E.: Adaptive detection of approximately duplicate database records and the database integration approach to information discovery. PhD Thesis, University of California, San Diego (1997)
33. OCLC. Persistent Uniform Resource Locator. Web page. http://purl.oclc.org/
34. Library of Congress. LC Digital Repository Development Core Metadata Elements Introduction Page. Web page, (2004) http://www.loc.gov/standards/metadata.html
35. On, B.-W., Elmacioglu, E., Lee, D., Kang, J., Pei, J.: An effective approach to entity resolution problem using quasi-clique and its application to digital libraries. In: ACM/IEEE Joint Conf. on Digital Libraries (JCDL), Jun (2006)
36. On, B.-W., Lee, D., Kang, J., Mitra, P.: Comparative study of name disambiguation problem using a scalable blocking-based framework. In: ACM/IEEE Joint Conf. on Digital Libraries (JCDL), Jun. (2005)
37. Paskin, N.: DOI: a 2003 Progress Report. D-Lib Magaz **9**(6) (2003)
38. Pasula, H., Marthi, B., Milch, B., Russell, S., Shpitser, I.: Identity uncertainty and citation matching. In: Advances in Neural Information Processing Systems. MIT Press, Cambridge (2003)
39. Petinot, Y., Teregowda, P.B., Han, H., Giles, C.L., Lawrence, S., Rangaswamy, A., Pal, N.: eBizSearch: An OAI-compliant digital library for ebusiness. In: ACM/IEEE Joint Conf. on Digital Libraries (JCDL), Houston, May (2003)
40. The Open Citation Project. http://opcit.eprints.org/
41. SecondString: Open source Java-based Package of Approximate String-Matching. http://secondstring.sourceforge.net/
42. Synman, M.M.M., van Rensburg, M.J.: Revolutionizing Name Authority Control. In ACM Int'l Conference on Digital Libraries (DL) (2000)
43. Takasu, A.: Bibliographic attribute extraction from erroneous references based on a statistical model. In: ACM/IEEE Joint Conf. on Digital Libraries (JCDL), Houston, May (2003)
44. Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Inf. Sys. **26**(8), 607–633 (2001)
45. Tillett, B.: FRBR: A conceptual model for the bibliographic universe. Library of Congress Cataloging Distribution Service, 2004. http://www.loc.gov/cds/downloads/FRBR.PDF.
46. VIAF. Virtual International Authority File (VIAF) project. Web page. http://www.oclc.org/research/projects/viaf/default.htm
47. Warnner, J.W., Brown, E.W.: Automated name authority control. In: ACM/IEEE Joint Conf. on Digital Libraries (JCDL) (2001)
48. Winkler, W.E.: The state of record linkage and current research problems. Technical report, US Bureau of the Census, Apr (1999)