System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach

Yoojin Hong[1], Byung-Won On[1], and Dongwon Lee[2]

[1] Department of Computer Science and Engineering, The Pennsylvania State University
{yohong, on}@cse.psu.edu

[2] School of Information Sciences and Technology, The Pennsylvania State University
dongwon@psu.edu

Abstract. In maintaining Digital Libraries, having bibliographic data up-to-date is critical, yet often minor irregularities may cause information isolation. Unlike documents for which various kinds of unique ID systems exist (e.g., DOI, ISBN), other bibliographic entities such as author and publication venue do not have unique IDs. Therefore, in current Digital Libraries, tracking such bibliographic entities is not trivial. For instance, suppose a scholar changes her last name from *A* to *B*. Then, a user, searching for her publications under the new name *B*, cannot get old publications that appeared under *A* although they are by the same person. For such a scenario, since both *A* and *B* are the same person, it would be desirable for Digital Libraries to track their identities accordingly. In this paper, we investigate this problem known as *name authority control*, and present our system-oriented solution. We first identify three core building blocks that underlie the phenomenon, and show taxonomy where different combinations of the building blocks can occur. Then, we consider how systems can support the problem in two common functions of Digital Libraries - *Update* and *Search*. Finally, our test-bed called OpenDBLP is presented where the suggested solution is fully implemented as a proof of the concept.

1. Introduction

A bibliographic Digital Library (DL) such as DBLP [3], CiteSeer [9] or e-Print arXiv [10] archives a collection of articles and their citation data in a certain domain. Often, such a DL is a starting place for researchers to locate relevant works, and a good test-bed for various citation analysis studies. A *citation* or *reference* consists of various bibliographic fields (e.g., author, title, conference/journal name or year), which we refer to as *Bibliographic Entity* in this paper. Often, documents have ways to track their identity. For instance, similar to the case where ISBN can serve as a unique ID for books, Digital Object Identifier (DOI) [11] can provide a persistent ID for digital documents. Therefore, even if two citations are slightly different in format, if their associated DOIs are identical, then one knows two citations in fact refer to the same object in the real world. As DLs evolve over the time, bibliographic entities change too. Especially, due to data-entry errors or different formats used in references, DLs

often contain a large variety of values referring to the same objects. For instance, Figure 1, inspired from [14], is the screen-shot of a search session in CiteSeer, looking for a book "Artificial Intelligence: A Modern Approach" by Russell and Norvig. Note that CiteSeer currently lists "23" citations as different, but all of them refer to the identical book, thus must be consolidated into a single citation.

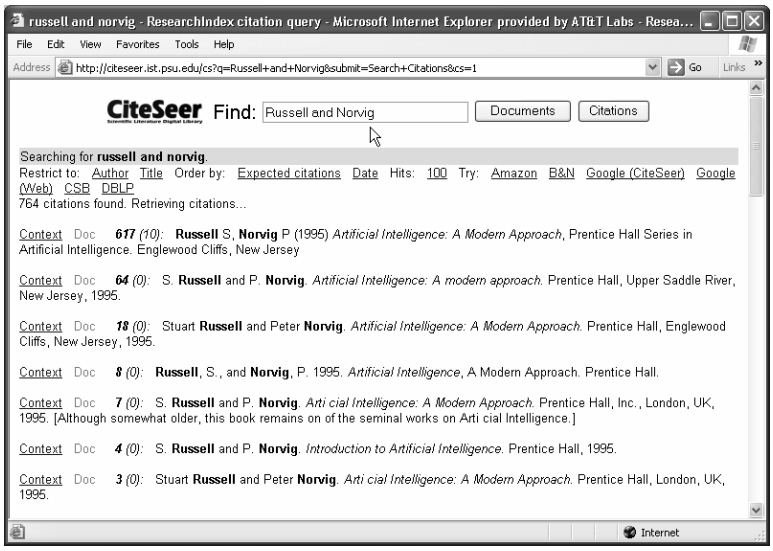

Figure 1. Search results showing name authority control problem.

This so-called *Citation Matching* problem (or more generally known as *Record Linkage* [13] or *Identity Uncertainty* [14]) is mainly due to different formats people use on the Web or in publications. Toward this problem, people have devised various methods to automatically detect duplicate or similar bibliographic entities that refer to the same object (e.g., [13][14][8]). For instance, using Levenstein edit distance or Jaro distance, one may detect that "S. Russell" and "Russell, Stuart" are the same person. However, to our best knowledge, there has been little work as to how systems can support updating and searching duplicates, once such matches are identified. Furthermore, previous research tends to focus on irregularities caused by errors (e.g., misspell, data-entry error). However, there are also "semantic" irregularities that are legitimate but unavoidable (e.g., a person changes last name after marriage). Therefore, in this paper, we investigate how to support maintenance of DLs once various semantic irregularities are identified. We especially focus on tracking two bibliographic entities – *author* and *publication venue*. First, let us consider various semantic irregularities of two entities that may occur in DLs:

- **Author**: Since the identity of a person is often determined by the name, if person's name changes over time, system cannot keep track of the person's bibliographic records uniformly. For instance, when an author "Alon Levy" becomes "Alon Halevy" after marriage, DLs view two authors as different

persons. Likewise, when two scholars share the same name, system cannot differentiate them. For instance, DBLP views two "Wei Wang"s, one at U. North Carolina and the other at HKUST but both are database researchers, as one person. Another case is that DL cannot recognize different varieties of a name. For instance, "Lee D. Coraor" and "Lee Coraor" are treated as two different persons although both refer to the same professor at Penn State.

- **Publication Venue**: Similarly, the identity of publication venue such as conference/journal/publisher is also determined by the name in DLs, but the name can be dynamic. For instance, multiple conferences may merge into a single conference over time (e.g., "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").

To handle such semantic irregularities, running citation matching methods periodically is one way. However, not only the accuracy of such methods is less than perfect, it is wasteful from a system point of view. Once DL learns that both "S. Russell" and "Russell, Stuart" are the same person, it is more desirable for the system to keep that knowledge to exploit it in future. Similarly, after DL learns that "ACM DL" and "IEEE ADL" were merged to "ACM/IEEE JCDL" in 2001, it may return publications from all three conferences for a query "find all publications in JCDL about Digital Identity after 1995" even if they are asked only to "JCDL."

2. Problem and Solution Overviews

We consider a problem as to how to track bibliographic entities in DLs, typically known as *Name Authority Control problem*. Formally, we solve the problem:

> *When bibliographic entities (i.e., author and publication venue) of citations change over time in Digital Libraries, devise a system support such that DLs can update and search the changes properly.*

Toward the problem, we first present three core elements – linear change, split, and merge – as basic building blocks, and discuss how systems can support those. More specifically, we discuss how UPDATE and SEARCH functions of DLs are changed to track bibliographic entities. Then, we present a proof of the concept, fully implemented in a test-bed, called OpenDBLP [1].

3. Related Work

Citation matching problem has been extensively investigated under various names in various disciplines. For instance, it bears a great relevance to problems known as

record linkage [13], identity uncertainty [14], merge-purge [2], etc. However, none of them concerns issues related to "system support" once matching citations are identified. Furthermore, we are interested in individual bibliographic entities – author and publication venue. Works done in [4][8] aim at detecting name variants automatically using data mining or heuristics techniques. Our work is complementary to them since we focus on system support issues once such variants are (semi-)automatically identified. Similarly, [5] 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). DOI [11] provide means to specify a persistent ID for digital objects. However their full acceptance is far from reality. Similarly, [6] discusses an effort to standardize author names using a unique number, called INSAN. [7] is a recent implementation for name authority control, called HoPEc, which bears some similarity to our approach. The detailed comparison between our OpenDBLP vs. [6][7] is summarized in Table 1.

Table 1. Comparison between OpenDBLP vs. INSAN [6], HoPEc [7].

	INSAN	**HoPEc**	**OpenDBLP**
Support for unique ID in the system?	Yes	No	Yes
Support for UPDATE?	Yes	Yes	Yes
Support for SEARCH?	No	No	Yes
Support for *Linear Change*?	Yes	Yes	Yes
Support for *Split*?	No	No	Yes
Support for *Merge*?	No	No	Yes

4. Our Approach

Although many variations seem to exist in the name authority control issues, at the bottom, there are only three core elements as follows:

1. Linear change (A→B). A bibliographic entity A is changed to B. For instance, an author or a conference/journal name is changed over the time.

2. Split ($A \rightarrow \{A_1, A_2\}$). A bibliographic entity is split into multiple ones. For instance, a conference can be broken into two or the publications of two scholars whose names have the same spelling can be split.

3. Merge ($\{A_1, A_2\} \rightarrow A$). Conversely, multiple bibliographic entities are merged into one. For instance, two variants of a person's name may be merged into one authoritative one.

We first discuss how two common functions of DLs, Update and Search, can be changed to support three core elements.

4.1 UPDATE Function

Once name variants are identified (manually by a librarian/author or automatically by algorithms), the findings must be inserted into DLs to solve the name authority

control. For instance, suppose an author "Wei Wang" realizes that her publication list is mixed with another person whose name has the same spelling as hers. Then, she may need to specify which of the publications in the DL belong to her and which does not. Or, if a librarian finds that publications under both "Lee Coraor" and "Lee D. Coraor" are by the same physical person, he probably wants to merge two publications lists into one by informing the DL all the name variants.

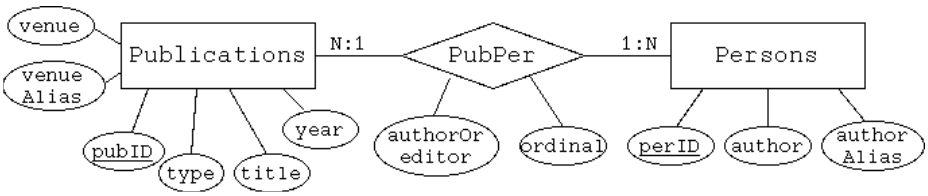

Figure 2. One possible ER diagram for bibliographic digital libraries.

Imagine a DL built on RDBMS with three tables shown in Figure 2: (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. 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*).

1. Linear change. When an entity 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 an entity 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 originally the publications of two "Wei Wang"s were incorrectly mixed, separating them out is the case of split as in WeiWang → {WeiWang, WeiWang}.

3. Merge. When name variants, A and B, are 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 so that when users want, he/she can still search using old name variants A and B.

4.2 SEARCH Function

Once duplicates are identified and put into the system, those knowledge can be exploited in searching. Suppose a user is looking at all publications about XML by "Alon Halevy," a database researcher at U. Washington, USA. When he submits two words "Alon Halevy" in the search box of DLs, internally, an SQL query similar to the following (assuming the schema of Figure 2) 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". When such related information is updated by the previous function, now the system can do 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 Halevy' AND P1.title LIKE '%XML%' AND
        P3.authorAlias = P4.author)
```

According to three core elements, over the time dimension, there are various strategies on how DLs can react to such a search function. Suppose conferences, C1 to C8, have evolved as follows: C1→C2 (i.e., name change), C3→{C4,C5} (i.e., conference split), and {C6,C7}→C8 (i.e., conference merge). Three possible strategies for searching conferences after the name evolutions are illustrated in Table 2. 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 C2, system shows C2 as well as all its predecessors, C1, as answers. Another possible strategy is the "*semantic*" search, where all semantically related results are returned, regardless of the temporal aspect. That is, the semantic strategy is equal to the union of both backward and forward strategies. For instance, since C3 is broken into C4 and C5, whenever browsing C3 occurs, it is expanded to all conferences related to C3, thus C4 and C5. Note, however, that browsing C4 is not expanded to C5.

Table 2. Various searching strategies when name authority control is considered.

Search	Return
C1	C1
C2	C1,C2
C3	C3
C4	C3,C4
C5	C3,C5
C6	C6
C7	C7
C8	C6,C7,C8

(a) Backward

Search	Return
C1	C1,C2
C2	C2
C3	C3,C4,C5
C4	C4
C5	C5
C6	C6,C8
C7	C7,C8
C8	C8

(b) Forward

Search	Return
C1	C1,C2
C2	C1,C2
C3	C3,C4,C5
C4	C3,C4
C5	C3,C5
C6	C6,C8
C7	C7,C8
C8	C6,C7,C8

(c) Semantic

4.3 Taxonomy of Name Authority Control

By combining the three core elements as basic building blocks, one can cover various real patterns found in most DLs. Suppose two elements can be concatenated by "*", the concatenation operator (e.g., *split * merge*). We have analyzed DBLP thoroughly and uncovered various cases where concatenations of two elements need to be supported. Table 3 shows the taxonomy, where (1) different alphabets (A, B, …) represent different name strings, (2) different subscripts (A_1, A_2, …) for the same alphabet represent name variants, and (3) **bold fonts** represent the active authoritive names (i.e., not an alias).

Table 3. Taxonomy of name authority control for three core elements.

Concatenation (*)	*Linear*	*Split*	*Merge*
Linear	Case 1: A→B→**C**	Case 2: A→B→{**B_1,**B_2}	Case 3: A→B {B,C}→**B**
Split	Case 4: A→{A_1,**A_2**} A_1→**B**	Case 5: A→{A_1,**A_2**}→{**A_{1a},A_{1b}**}	Case 6: A→{A_1,A_2} {A_1,B}→**A_1**
Merge	Case 7: {A,B}→A→**C**	Case 8: {A,B}→A→{**A_1,**A_2}	Case 9: {A,B}→A {A,C}→**A**

*Case 1 (Linear * Linear).* An author happens to change his name twice over the time. After applying a solution of *Linear* twice, a name A is changed to C.

*Case 2 (Linear * Split).* An author changes name from A to B, but there is the same name B already, mixing two publications together. To avoid the mixture, *split* is applied to B, yielding two name variants B_1 and B_2.

*Case 3 (Linear * Merge).* At time t_0, an author used A as the name, but later at time t_1, he changed A to B. However, due to data-entry error, C was used to refer to B as well. To fix this, *merge* is applied to B and C, making B as the authoritative name.

*Case 4 (Split * Linear).* An author A_1 wants to change his name to B, but currently his records are mixed with another name A_2.

*Case 5 (Split * Split).* When three authors' publications are mixed under the one name, A, then two *split* operations are applied in a row.

*Case 6 (Split * Merge).* An author A_1 found that his publication records are mixed with other name A. Also, he also found that B is his alias. Therefore, he wants to separate his records from A first, and consolidate them with B's records. Furthermore, he likes to use A_1 as the authority.

*Case 7 (Merge * Linear).* An author A wants to merge his publications registered with his name variety B to his. Also, he wants to change his name to C. Since the name A

and B cannot be changed to completely different name C at once by using 'Merge my profiles', they firstly merged to the name A. Then, the name A can be the name C.

*Case 8 (Merge * Split).* At time t_0, two conference names, A and B, are merged into A, but later at time t_1, A is changed to C.

*Case 9 (Merge * Merge).* When an author has many name variants (e.g., "Jeffrey Ullman," "J. Ullman," and "Ullman, Jeffrey D."), multiple consecutive *merge* operations can be applied.

5. System Implementation

5.1 Overview of the OpenDBLP

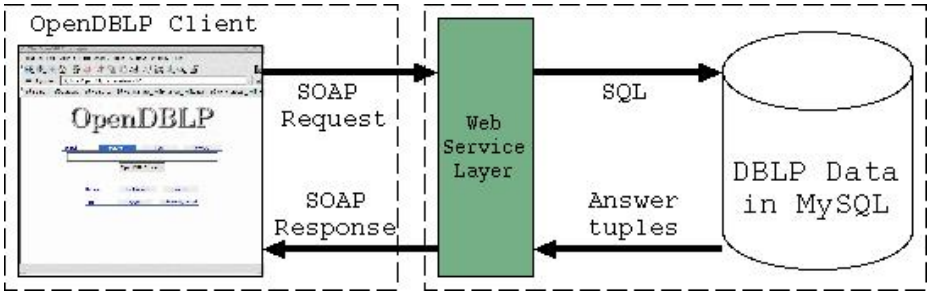

Fig 3. The OpenDBLP system.

Since the proposed solution is implemented in a test-bed, called OpenDBLP, in this section, we give a brief overview, as shown in Figure 3. The OpenDBLP is a rejuvenated version of the popular DBLP digital library with a few novel improvements: (1) fully DBMS-based storage system, supporting ranked and approximate query processing, (2) web service based programmable interface (box in Fig. 3) to the contents of DBLP, and (3) a web client program that faithfully mimics the original FORM interface of the DBLP. Especially, this program fully implements old "Browse" and "Search" interfaces using only web services.

The prototype system is accessible at **http://opendblp.psu.edu/**.

5.2 UPDATE Function in OpenDBLP

Here, we briefly show how three core elements are implemented in OpenDBLP. Users can update his/her records (after manually or automatically finding some semantic irregularities) using one of the tree menus shown in Fig 4.

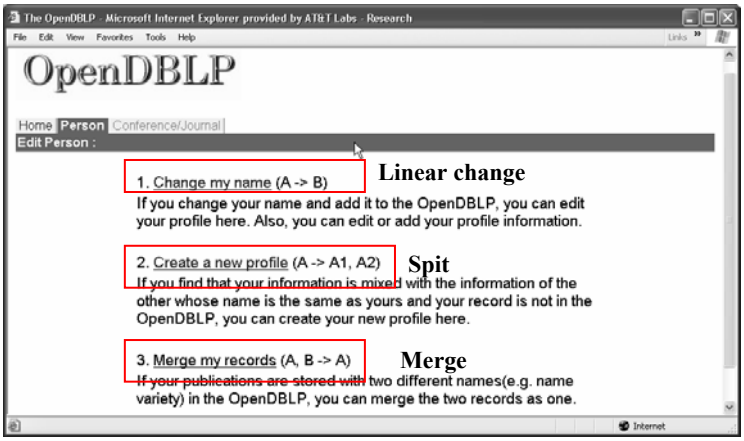

Fig 4. OpenDBLP UPDATE menus

Figures 5 – 7 demonstrate (1) the "Alon Levy" case of *linear change*, (2) "Wei Wang" case of *split*, and (3) "Lee Coraor" case of *merge*, respectively.

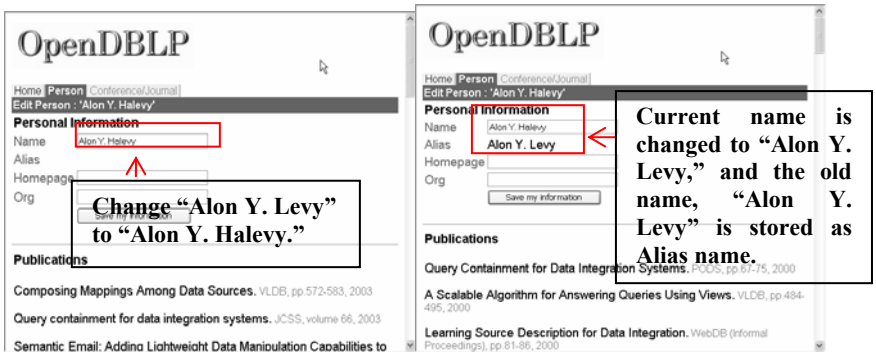

Fig 5. Changing a name for "Alon Levy": before (left) and after (right).

For the name change of "Alon Levy" as shown in Fig 5, the person name in *Person* table is changed to the new name and the old name is stored as an alias. Note that it is important to keep old records of "Alon Levy" although his current name is "Alon Halevy" for historical purposes – some users may ask queries specifically using his old name.

Fig 6. Creating a new profile for "Wei Wang": before (left) and after (right).

For spliting bibliographic information of "Wei Wang" in HKUST, OpenDBLP creates a new record in *Person*s table and assigns a new perID. Then, all the right publications (checked by users through web interface) are carried along to the newly generated perID. At the end, there are two "Wei Wang"s in the system, each kept separately and recognized as different despite their same spelling by the system.

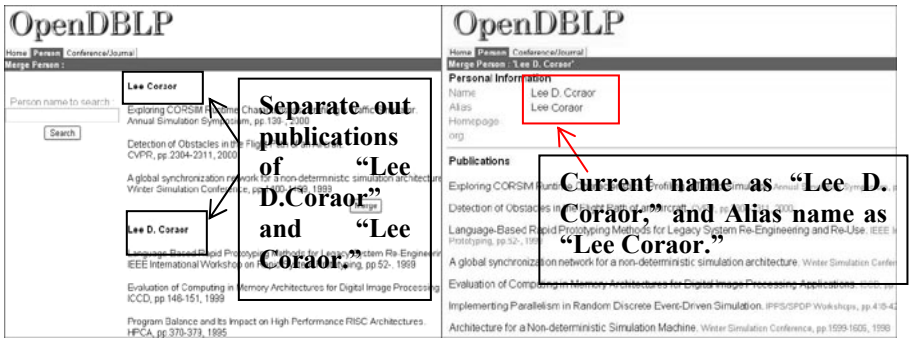

Fig 7. Merging two records for "Lee Coraor": before (left) and after (right).

For the third example of "Lee Coraor", OpenDBLP chooses one of the names as his current name (i.e., the authoritive one) and stores another name as an alias. The perID for all his publications in *PubPer* table is re-written to the perID of the chosen name.

5.3 SEARCH Function in OpenDBLP

Once updates are successfully made, in searching, OpenDBLP can exploit its knowledge on name authority control using one of the strategies in Section4.2. Figures 8 – 10 illustrates the improved search in OpenDBLP that can automatically show, for instance, current as well as "old-but-relevant" authors or publication venues.

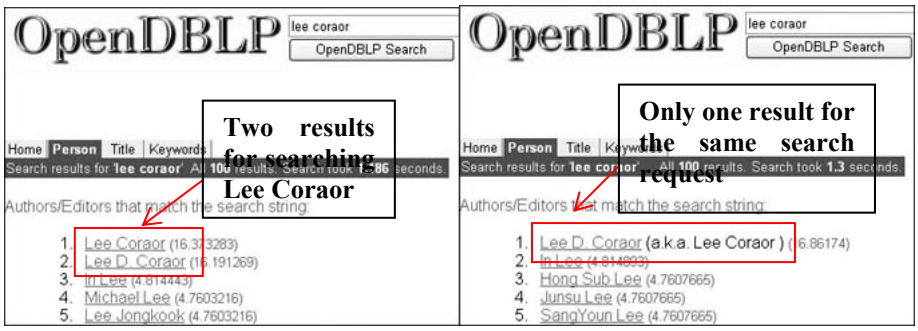

Fig 8. Search result for "Lee Coraor" after update: before (left) and after (right).

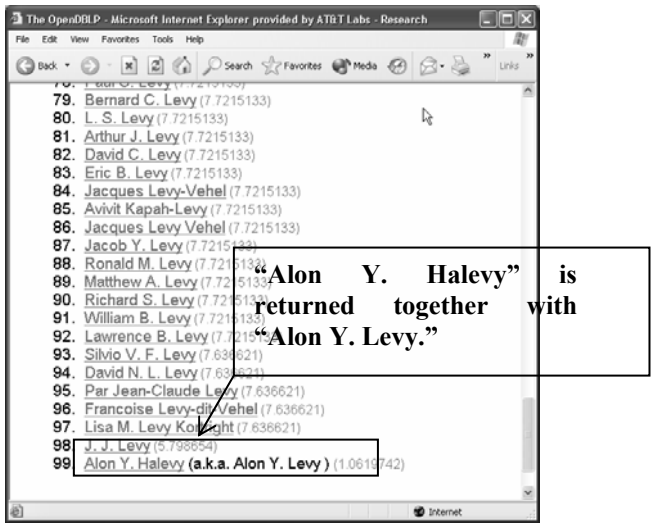

Fig 9. Search result for "Levy" after update.

Fig 10. Search result for "Wei Wang" after update: before (left) and after (right).

6. Conclusion

In the paper, we re-visited the *name authority control problem* in Digital Libraries (DL) and presented a solution that is, unlike the previous approaches, more focused on the system-support issues of how DLs can effectively support the problem in (1) *updating* records with name authority control problem, and (2) *searching* related records by exploiting the knowledge about name authority control. We have identified that three are mainly three core elements that lie in most name-related changes as follows: (1) *linear change* of entity from A to B; (2) *split* of an entity A to multiple entities; (3) *merge* of multiple entities into single one. Using different combinations of these core elements, we have shown that many of the name authority problems can be expressed and thus solved. Finally, all the proposed solutions are fully implemented in a test-bed, OpenDBLP system, so that two of the common functions of DLs, Update and Search, can fully track down the right bibliographic entities despite the usage of different values. Although our proposed solution was tested only on a particular domain (i.e., DBLP), the techniques that we have developed can be applied to other DLs in a straightforward manner.

References

[1] Y. Hong and D. Lee. “OpenDBLP: Rejuvenating the DBLP into Web Service Based Programmable Digital Library.” Technical report, Penn State University, 2004

[2] M. A. Hernandez and S. J. Stolfo. “The Merge/Purge Problem for Large Databases,” ACM SIGMOD, 1995.

[3] M. Ley. “The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives”. SPIRE, Lisbon, Portugal, Sep. 2002.

[4] J.W. Warnner and E.W. Brown. “Automated Name Authority Control”. ACM/IEEE JCDL, 2001.

[5] P. T. Davis, D. K. Elson, and J. L. Klavans. “Methods for Precise Named Entity Matching in Digital Collections”. ACM/IEEE JCDL, 2003.

[6] M. M. M. Synman and M. J. van Rensburg. “Revolutionizing Name Authority Control”. ACM DL, 2000.

[7] J. M. B. Cruz, N. J. R. Klink, and T. Krichel. “Personal Data in a Large Digital Library”. ECDL, 2000.

[8] H. Han, C. L. Giles, H. Zha et al. "Two Supervised Learning Approaches for Name Disambiguation in Author Citations," ACM/IEEE JCDL, 2004.

[9] CiteSeer: Scientific Literature Digital Library, http://citeseer.ist.psu.edu/

[10] arXiv.org e-Print archive, http://arxiv.org/

[11] H. Atkins, C. Lyons, H. Ratner, C. Risher, C. Shillum, D. Sidman, A. Stevens, and W. Arms. “Reference Linking with DOIs: A Case Study," D-Lib Magazine, 2000.

[12] The Open Citation Project. http://opcit.eprints.org/

[13] I. P. Fellegi and A. B. Sunter. “A Theory for Record Linkage,” J. of the American Statistical Society, 64:1183-1210, 1969.

[14] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. “Identity Uncertainty and Citation Matching,” Advances in Neural Info. Processing Sys. MIT Press, 2003.