Effective and Scalable Solutions for Mixed and Split Citation Problems in Digital Libraries

Dongwon Lee[*] — Penn State / USA — dongwon@psu.edu
Byung-Won On — Penn State / USA — on@cse.psu.edu
Jaewoo Kang — NCSU / USA — kang@csc.ncsu.edu
Sanghyun Park — Yonsei Univ. / Korea — sanghyun@cs.yonsei.ac.kr

ABSTRACT

In this paper, we consider two important problems that commonly occur in bibliographic digital libraries, which seriously degrade their data qualities: *Mixed Citation (MC)* problem (i.e., citations of different scholars with their names being homonyms are mixed together) and *Split Citation (SC)* problem (i.e., citations of the same author appear under different name variants). In particular, we investigate an effective yet scalable solution since citations in such digital libraries tend to be large-scale. After formally defining the problems and accompanying challenges, we present an effective solution that is based on the state-of-the-art sampling-based approximate join algorithm. Our claim is verified through preliminary experimental results.

1. INTRODUCTION

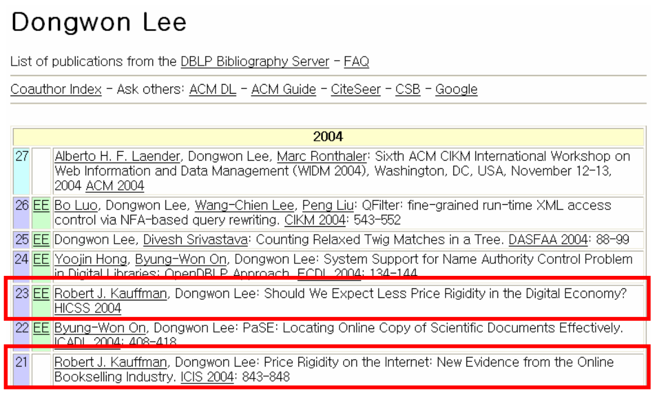
Bibliographic Digital Libraries (DLs), such as DBLP, CiteSeer or e-Print arXiv, contain a large number of citation[1] records. Such DLs have been an important resource for academic communities since scholars often try to search for relevant works from DLs. Researchers also use the citation records in order to measure the publication's impact in the research community. In addition, citations are often used when users search for articles of interest. Therefore, it is important to keep the citations of DLs consistent and up-to-date. However, due to data entry errors, imperfect citation gathering software or common author names, DLs often have many errors in their citation collections. In particular, we focus on two problems that commonly occur in many existing DLs as follows.

First, when two scholars have the same name spellings, their citation data are mistakenly merged into a single collection, leading to an incorrect citation analysis results. We call this as **Mixed Citation (MC)** problem. For instance,

[*]The author is partially supported by IBM Eclipse Innovation Award (2004) and Microsoft SciData Award (2005).

[1]A *citation* or *reference* is a bibliographic entity that consists of various fields (e.g., author, title, venue or year).

IQIS 2005, June 17, 2005, Baltimore, MD, USA.



Dongwon Lee

List of publications from the DBLP Bibliography Server - FAQ

Coauthor Index - Ask others: ACM DL - ACM Guide - CiteSeer - CSB - Google

2004		
27		Alberto H. F. Laender, Dongwon Lee, Marc Ronthaler: Sixth ACM CIKM International Workshop on Web Information and Data Management (WIDM 2004), Washington, DC, USA, November 12-13, 2004 ACM 2004
26	EE	Bo Luo, Dongwon Lee, Wang-Chien Lee, Peng Liu: QFilter: fine-grained run-time XML access control via NFA-based query rewriting. CIKM 2004: 543-552
25	EE	Dongwon Lee, Divesh Srivastava: Counting Relaxed Twig Matches in a Tree. DASFAA 2004: 88-99
24	EE	Yoojin Hong, Byung-Won On, Dongwon Lee: System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach. ECDL 2004: 134-144
23	EE	Robert J. Kauffman, Dongwon Lee: Should We Expect Less Price Rigidity in the Digital Economy? HICSS 2004
22	EE	Byung-Won On, Dongwon Lee: PaSE: Locating Online Copy of Scientific Documents Effectively. ICADL 2004: 408-418
21		Robert J. Kauffman, Dongwon Lee: Price Rigidity on the Internet: New Evidence from the Online Bookselling Industry. ICIS 2004: 843-848

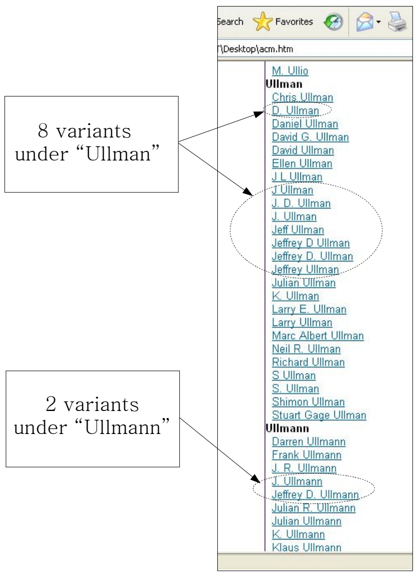
Figure 1: Mixed citations of "Dongwon Lee" in DBLP. Boxed ones are by another "Dongwon Lee."

Figure 1 illustrates a collection of citation data by one of the authors, "Dongwon Lee", in DBLP. Note that two citations by "another" scholar with the same name spelling are listed (boxed ones). The reason of this mixture is that there exist two scholars with the name "Dongwon Lee" – a computer scientist at Penn State and an MIS scholar at U. Minnesota – with somewhat overlapping domains of interests.

Second, due to various reasons, DLs tend to keep the citations of single author under various *name variants*[2]. We call this as **Split Citation (SC)** problem. For instance, imagine a scholar "John Doe" has published 100 articles. However, a DL keeps two separate name variants, "John Doe" and "J. D. Doe", each of which contains 50 citations. In such a case, users searching for all the articles of "John Doe" will get only 50 of them. Similarly, any bibliometrical study would underestimate the impact of the author "John Doe", splitting his fare share into "John Doe" and "J. D. Doe" incorrectly. Such a problem of ambiguous author names exists in many of existing DLs, as illustrated in Figure 2, where a renowned computer scientist "Jeffrey D. Ullman" appear under 10 different name variants in ACM Portal's DL.

In essence, both MC and SC problems cannot be completely avoided unless each person carries a universal ID. Note that these problems would still occur even if digital

[2]In this paper, the *name variants* refer to the different spellings of author names that are in fact referring to the same person.



M. Ullio
Ullman
Chris Ullman
D. Ullman
Daniel Ullman
David G. Ullman
David Ullman
Ellen Ullman
J L Ullman
J Ullman
J. D. Ullman
J. Ullman
Jeff Ullman
Jeffrey D Ullman
Jeffrey D. Ullman
Jeffrey Ullman
Julian Ullman
K. Ullman
Larry E. Ullman
Larry Ullman
Marc Albert Ullman
Neil R. Ullman
Richard Ullman
S Ullman
S. Ullman
Shimon Ullman
Stuart Gage Ullman
Ullmann
Darren Ullmann
Frank Ullmann
J. R. Ullmann
J. Ullmann
Jeffrey D. Ullmann
Julian R. Ullmann
Julian Ullmann
K. Ullmann
Klaus Ullmann

Figure 2: Split citations of "Jeffrey D. Ullman" in ACM Portal.

object identifier (DOI)[3] system is fully adopted, since it usually does not govern the identity of a person or her name. In this paper, therefore, we investigate efficient solutions for these problems.

2. BACKGROUND

Related Work. In [14], we investigated issues related to system support for both problems, and in [20], we explored the split citation problem. Han et al. [12] proposed two supervised learning-based approaches for a related problem. Their problem can be viewed as a kind of the SC problem in our jargon although they do not explicitly define the problem. Furthermore, their approach is not scalable to handle large-scale DLs. On the other hand, we investigate both the MC and SC problems, and present scalable solutions. Nevertheless, we also tested their ideas of supervised learning methods in the step 2 of the name disambiguation algorithm (Section 4.2). The goal of our study is not to compare supervised methods against unsupervised ones, but to explore the combinations of alternatives that give good scalability/accuracy trade-offs in the two-step approaches. We recently learned that [2] introduces a clustering-based solution to a problem similar to our MC problem, and the performance comparison is currently underway. ALIAS system in [22] proposes a framework to detect duplicate entities such as citations or addresses, but its focus is on the learning aspect.

Since the core technique of our proposals in this paper is to match two related citations (i.e., citation matching), our work is closely related to more general class of problems, known as various names – record linkage (e.g., [10, 3]), citation matching (e.g., [19]), identity uncertainty (e.g., [21]), merge-purge (e.g., [13]), object matching (e.g., [5]), duplicate detection (e.g., [22, 1]), approximate string join (e.g., [11]) etc.

String similarity measures used in our work were proposed by Jaro [16] and Winkler [25]. Bilenko et al. have studied name matching for information integration [3] using string-based and token-based methods. Cohen et al. have also compared the efficacy of string-distance metrics, like JaroWinkler, for the name matching task [6]. In DLs, this problem is called citation matching. In the citation matching domain, [18] experimented with various distance-based algorithms with a conclusion that word based matching performs well. We have implemented all these methods in the second step of your algorithm and compared their efficacy to other methods.

Before we can process citations, we assume that field segmentation and identification has been completed using some methods like one in [4]. Blocking was first proposed by Kelley et al. [17] in the record linkage literature. Our sampling idea can be viewed as a blocking scheme and is also similar in flavor to the two-step citation matching schemes proposed in [15, 19] where initial rough but fast clustering (or "Canopy") is followed by more exhaustive citation matching step.

Another stream of works that are relevant to our work is name/entity disambiguation and authority controls in NLP community. For instance, works done in [24] aim at detecting name variants automatically using data mining or heuristics techniques, but do not consider the issue of scalability nor in the context of digital libraries. Similarly, [9] 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). [23] discusses an effort to standardize author names using a unique number, called INSAN, and [8] is a recent implementation for name authority control, called HoPEc. On the contrary, we focus more on two specific problems relevant to citations of digital libraries.

Preliminaries. In this Section, we introduce a technique that our solutions exploit. One of the state-of-the-art sampling techniques that satisfy both criteria (i.e., being fast and accurate) is the *sampling-based join approximation* method recently proposed by [11]. We adopt it to our context as follows: Their main idea is that if, for each string n_i, one is able to extract a small sample S that contains mostly strings suspected to be highly similar to n_i, then this sample S serves as a candidate set, and the remaining strings can be quickly ignored (i.e., pre-filtering). To get the "good" sample S, imagine each token from all strings has an associated weight using the TFIDF metric in IR (i.e., common tokens in strings have lower weights while rare ones have higher weights). Then, each string t is associated with its token weight vector v_t. Suppose that, for each string t_q in a string set R_1, we want to draw a sample of size S from another string set R_2 such that the frequency C_i of string $t_i \in R_2$ can be used to approximate $sim(v_{t_q}, v_{t_i}) = \sigma_i$. That is, σ_i can be approximated by $\frac{C_i}{S} T_V(t_q)$, where $T_V(t_q) = \sum_{i=1}^{|R_2|} \sigma_i$. Then, put t_i into a candidate set only if $\frac{C_i}{S} T_V(t_q) \geq \theta$, where θ is a pre-determined threshold[4]. This strategy assures that all pairs of strings with similarity of at least θ survive the pre-filtering stage and put into the candidate set with a desired probability, as long as the proper sample size S is given.

[3]http://www.doi.org/

[4]In experimentation, we used the more optimized version of the sampling-based join approximation with a single scan from [11].

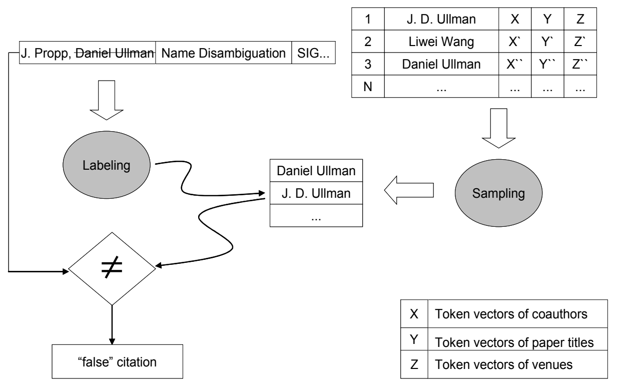


Figure 3: Overview of our solution to MC problem.

3. THE MIXED CITATION PROBLEM

3.1 Problem Definition & Solution Overview

Problem Definition. We formally define the *Mixed Citation* problem as follows:

> *Given a collection of citations, C, by an author, a_i, can we quickly and accurately identify false citations by another author a_j, when a_i and a_j have the identical name spellings?*

The challenge here is that since two different authors, a_i and a_j, have the "same" name spellings, one cannot easily distinguish the two by using distance between their names (e.g., Edit distance). To overcome this difficulty, we propose to exploit author's associated information. That is, given an author a_i, we may use additional information such as her coauthor list, common keywords that she often use in the titles of articles, or common publication outlets, etc.

Solution Overview. Consider a citation c_i with a set of coauthors $A = \{a_1, ..., a_n\}$, a set of keywords from the title $T = \{t_1, ..., t_m\}$, and a venue name V. Then, after removing the i-th coauthor a_i ($\in A$), can we correctly label c_i to a_i? That is, when a_i is removed from the citation c_i, can we guess back the removed author using associated information? Let us call this method as *Citation Labeling* algorithm. If we assume that there is a "good" citation labeling function $f_{cl} : c_i \rightarrow a_j$. Then, using the f_{cl}, the original MC problem can be solved as follows. Given citations C by an author a_1:

```
for each citation $c_i$ ($\in C$)
    remove $a_1$ (i.e., original name) 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 $c_i$ is a false citation;
    remove $c_i$ from $C$;
```

At the end, C has only correct citations by a_1. Therefore, if one can find a good citation labeling function f_{cl}, then one can solve the MC problem.

3.2 Citation Labeling Algorithm

Let us examine f_{cl} more closely. 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 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 $\phi$;
```

This baseline approach presents two technical challenges: (1) Since c_i and a_j are two different entities to compare in real world, 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 $O(|C||A|)$ is prohibitively expensive to run (e.g., the DBLP has about 0.56 million authors). In order to address these challenges, we propose two solutions as follows.

Similarity between Citation and Author. In [12], authors reported a promising result by representing a citation as 3-tuple of coauthors, titles, and venues. Although proposed for a different problem, the idea of 3-tuple representation of citations can be adapted to our context as follows: 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:

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

where $\alpha + \beta + \gamma = 1$ (i.e., weighting factors), $\vec{c_c}$, $\vec{c_t}$, and $\vec{c_v}$ are token vectors of coauthors, paper titles, and venues, respectively, of the citation c, and $\vec{a_c}$, $\vec{a_t}$, and $\vec{a_v}$ are token vectors of coauthors, paper titles, and venues from "all" citations of the author a, respectively. In turn, each similarity measure between two token vectors can be estimated using the standard IR techniques such as the *cosine similarity*, $\cos(\theta) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \cdot \|\vec{w}\|}$, along with TFIDF.

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: $\vec{c_c}$ = ["E.F. Codd"], $\vec{c_t}$ = ["Relational", "Model", "Data", "Large", "Shared", "Data", "Banks"], and $\vec{c_v}$ = ["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/IEEE Joint C. on Digital Libraries, 2005") is represented as: $\vec{a_c}$=["John Doe", "John Smith", "Dongwon Lee", "Jaewoo Kang"], $\vec{a_t}$=["Data", "Quality", "Algorithm", "Cleaning", "XML"], and $\vec{a_v}$=["IQIS", "ACM/IEEE", "Joint", "C.", "Digital", "Libraries"]. In Section 5, we study the variance of handling duplicate tokens (in set and bag models). 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 $sim(c, a)$.

Speed-up through Sampling. In general, the baseline approach has a quadratic time complexity which is too expensive for large-size DLs. However, note that for a citation c, one does not need to check if c can be labeled as an author a for all authors. If one can quickly determine candidate author set from all authors (i.e., pre-filtering), then c better be tested against only the authors in candidate set. We use the Gravano et al.'s approximate join algorithm introduced in Section 2 for the pre-filtering. That is,

```
for each citation $c_i$ ($\in C$)
```

[5]We pre-prune all stopwords from the title.

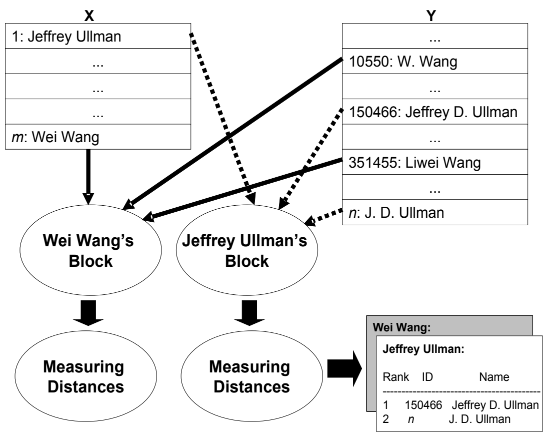


Figure 4: Overview of our solution to SC problem.

```
    draw a sample set $S(\subseteq A)$;
    examine all names $s_j (\in S)$;
    return $s_j$ $(\in S)$ with MAX $\phi$;
```

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

4. THE SPLIT CITATION PROBLEM

4.1 Problem Definition & Solution Overview

We formally define the *Split Citation* problem 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, ..., y_n $(\in Y)$.

The baseline approach to solve the problem is to treat each author name as a "string", and perform all pair-wise string distance using some distance function, $dist(x, y)$:

```
for each name $x$ $(\in X)$
  for each name $y$ $(\in Y)$
    if $dist(x, y) < \phi$, $x$ and $y$ are name variants;
```

This baseline approach has the limitations similar to the baseline approach of the MC problem in Section 3. 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, many authors from similar cultural or national background shares similar spellings in their name, those algorithms should not be too much dependent on the syntactic similarities of author-name strings.

Solution Overview. Figure 4 illustrates our name disambiguation algorithm: (1) Instead of syntactically comparing two name spellings alone, we use information associated with the author-name strings like coauthor list, authors' paper titles, and venue list. For instance, to identify if "Dongwon Lee" is the name variant of "D. Lee", instead of computing the string edit distance of two names, we may test if there is any correlation between the coauthor lists of "Dongwon Lee" and "D. Lee." In the remainder of the paper, we only focus on exploiting coauthor information as the only associated information of an author. Exploiting other associated information (or even hybrid of them as in [12]) is an interesting direction for future work; (2) To make the algorithm scalable, we again borrow the sampling idea.

Method		Step 1	Step 2
naive	1-N	–	name
two-step name-name	2-NN	name	name
two-step name-coauthor	2-NC	name	coauthor
two-step name-hybrid	2-NH	name	hybrid

Table 1: Solution space of name disambiguation algorithm.

4.2 Name Disambiguation Algorithm

The name disambiguation algorithm is as follows:

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

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

Depending on the choices in both Step 1 and 2, four variations of the name disambiguation algorithm are feasible, as summarized in Table 1: (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 the main proposal of ours 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., we used 1/4 and 3/4 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 author is represented as a potentially very long coauthor list, different distance metrics tend to show different accuracy/performance trade-off. To examine this issue, we have considered two supervised methods (i.e., Naive Bayes Model and Support Vector Machine) and five unsupervised methods (i.e., cosine, TFIDF, Jaccard, Jaro and JaroWinkler). In what follows, we briefly describe each method.

Naive Bayes Model (NBM). In this method, we use Bayes' Theorem to measure the similarity between two author names. For instance, to calculate the similarity between "Dongwon Lee" and "Lee, D.", we estimate the probability per coauthor of "Dongwon Lee" in terms of the Bayes rule in training, and then calculate the posterior probability of "Lee, D." with the coauthors' probability values of "Dongwon Lee" in testing. As shown in Figure 4, given a block corresponding to an author name x in X with the associated author names y_i in Y $(i \in [1, k]$, where k is the total number of author names from Y), we calculate the probability of each pair of x and y_i and find the pair with the maximal posterior probability as follows:

For training, a collection of coauthor names of x are randomly split, and only the half is used for training. We estimate each coauthor's conditional probability $P(A_m|x)$

Name	Description
x, y	coauthor names
T_x	all tokens of the coauthor x
C_x	all characters of x
$CC_{x,y}$	all characters in x common with y
$X_{x,y}$	# of transpositions of char. in x relative to y

Table 2: Terms.

conditioned on the event of x from the training data set, $A_i \in \{A_1, ..., A_j, ..., A_m\}$ and A_j is the j-th coauthor of x:

$$\begin{aligned} P(A_j|x) &= P(A_j|Frequent, Coauthor, x) \times \\ &\quad P(Frequent|Coauthor, x) \times P(Coauthor|x) + \\ &\quad P(A_j|Infrequent, Coauthor, x) \times \\ &\quad P(Infrequent|Coauthor, x) \times P(Alone|x) \end{aligned}$$

where $P(Alone|x)$ is the probability of x writing a paper alone, $P(Coauthor|x)$ is the probability of x working for a paper with coauthors in future, $P(Frequent|Coauthor, x)$ is the probability of x working for a paper with the coauthors, who worked with x at least twice in the training data, conditioned on the event of x's past coauthors, and $P(A_j|Frequent, Coauthor, x)$ is the probability of x working for a paper with a particular coauthor A_j

For testing, we use the following target function: $V_{NBM} = MAX_{y_i \in N}\{P(y_i)\Pi_k P(A_k|y_i)\}$, where N denotes is the total number of author names from Y in the block and A_k is the k-th coauthor in y_i, being the same coauthor as in x.

Support Vector Machines (SVM) is one of the popular supervised classification methods. In our context, it works as follows: First, all coauthor information of an author in a block is transformed into vector-space representation. Author names in a block are randomly split, and 50% is used for training, and the other 50% is used for testing. Given training examples of author names labeled either YES (e.g., "J. Ullman" and "Jeffrey D. Ullman") or NO (e.g., "J. Ullman", "James Ullmann"), the SVM creates a maximum-margin hyperplane that splits the YES and NO training examples. In testing, given the SVM classifies vectors by mapping them via kernel trick to a high dimensional space where the two classes of equivalent pairs and different ones are separated by a hyperplane, and the corresponding similarity is obtained. For the SVM prediction, we use the Radial Basis Function (RBF) kernel [7], $K(x_i, y_i) = e^{-\gamma||x_i - y_i||^2}, (\gamma > 0)$, among alternatives (e.g., linear, polynomial, sigmoid kernels).

String-based Distance Metrics. In this scheme, the distance between two author names are measured by the distance between their coauthor lists (thus no training is needed). That is, to measure the distance between "Dongwon Lee" and "Lee, D.", instead of computing *dist*("Dongwon Lee", "Lee, D."), we compute *dist*(coauthor-list("Dongwon Lee"), coauthor-list("Lee, D.")). Among many possible distance measures, we used two token-based string distances (e.g., *Jaro* and *TFIDF*), and two edit-distance-based ones (e.g., *Jaro* and *JaroWinkler*) that were reported to give a good performance for the general name matching problem in [6]. We briefly describe the metrics below. For details of each metric, refer to [6].

Using the terms of Table 2, the four metrics can be defined as follows: (1) $\mathbf{Jaccard(x,y)} = \frac{|T_x \bigcap T_y|}{|T_x \bigcup T_y|}$; (2) $\mathbf{TFIDF(x,y)} = \sum_{w \in T_x \cap T_y} V(w, T_x) \times V(w, T_y)$, where $V(w, T_x) = log(TF_{w,T_y} + 1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'}(log(TF_{w,T_y}+1) \times log(IDF_w))}}$ (symmetrical for $V(w, T_y)$), where TF_{w,T_x} is the frequency of w in T_x, and IDF_w is the inverse of the fraction of names in a corpus containing w; (3) $\mathbf{Jaro(x,y)} = \frac{1}{3} \times (\frac{|CC_{x,y}|}{|C_x|} + \frac{|CC_{y,x}|}{|C_y|} + \frac{|CC_{x,y}| - X_{CC_{x,y},CC_{y,x}}}{2|CC_{x,y}|})$; (4) $\mathbf{JaroWinkler(x,y)} = Jaro(x,y) + \frac{max(|L|,4)}{10} \times (1 - Jaro(x,y))$, where L is the longest common prefix of x and y.

Vector-based Cosine Distance. In this approach, instead of using string distances, we use vector distances to measure the similarity of the coauthor lists. We model the coauthor lists as vectors in the vector space, each dimension of which corresponds to a unique author name appearing in the citations. To measure the distance between two vectors, v and w, we use the simple cosine distance, an angle between two vectors, defined as: $\cos\theta = \frac{v \cdot w}{\|v\| \cdot \|w\|}$.

5. EXPERIMENTAL VALIDATION

5.1 Data Sets and Platform

We have gathered real citation data from four different domains, as summarized in Table 3. Compared to previous work, all of the four data sets are substantially "large-scale" (e.g., DBLP has 360K authors and 560K 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 [11] with a sample $S = 64$ and a threshold $\theta = 0.1$. For the 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[6] was used. For the string-based distance functions of the unsupervised learning methods, we used the implementations of TFIDF, Jaccard, Jaro, and JaroWinkler from SecondString[7]. Other remaining methods were implemented by us in Java. All experimentation was done using Microsoft SQL Server 2000 on Pentium III 3GHZ/512MB.

5.2 Results for MC Problem

Configuration. For this MC problem, we used two DLs out of four (due to time constraint) as test-beds: DBLP and EconPapers. For DBLP (which authors know well), we collected real examples with the MC problem: e.g., Dongwon Lee, Chen Li, Wei Liu, Prasenjit Mitra, and Wei Wang, etc, and for EconPapers (which authors do not know well), we injected an artificial "false citations" into each author's citation collection. 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

[6]http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[7]http://secondstring.sourceforge.net/

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

Table 3: Summary of data sets.

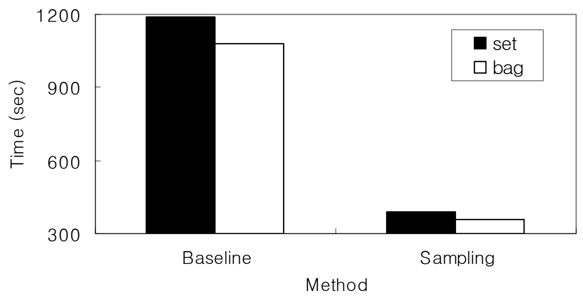


Figure 5: Scalability (EconPapers).

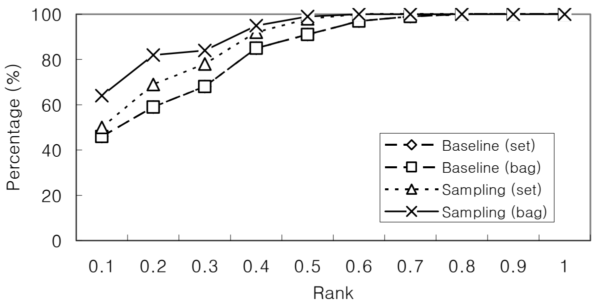
the preservation of the multiple occurrences of the same token. For testing, we used the weights, $\alpha = 0.5$, $\beta = 0.3$, and $\gamma = 0.2$. As evaluation metrics, we used *time* for scalability, and *percentage/rank* ratio for accuracy (i.e., A false citation c_f must be ranked low in $sim(c_f, a)$. Thus, we measured how much percentage of false citations were ranked in the bottom 10%, 20%, etc).

Results. First, Figure 5 clearly shows the superior scalability of the sampling-based approach over the baseline one (about 3-4 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, Figure 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). Furthremore, 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, Figure 6(b) shows results on DBLP using only the bag model. 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"). 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. We leave a solution to remedy this problem as a future work.

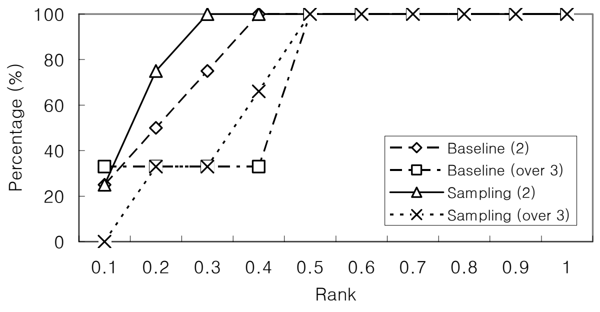
5.3 Results for SC Problem

Configuration. Ideally, it would be desirable to apply our framework to existing DLs to find all real name variants. However, given a large number of citations that we aim at, it is not possible nor practical to find a "real" solution set. For instance, to determine if "A" is indeed a name variant of "B", human experts have to trace it carefully. Therefore, here, we use synthetic solution sets. Nevertheless, in practice, we envision that our framework be used as a tool to assist human experts to narrow down candidate name variants significantly.

To make solution sets, for each data set, we prepare two initially-empty lists, X and Y. Then, we pick 100 authors with substantial number of citations (so that supervised methods can be trained), and put them into X. For each of 100 original names, x_n, in addition, we create an artificial name variant, y_n. Furthermore, citations of each x_n are randomly split into two sets, and assigned to x_n and y_n, respectively (i.e., each name carries half of the original citations). Finally, to make the disambiguation more challenging, we dump the entire author names into Y. For instance, for DBLP test case, there are 100 and 364,377 names in X and Y, respectively. Then, through the proposed two-step name 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 it is).



(a) EconPapers



(b) DBLP

Figure 6: Accuracy (EconPapers and DBLP).

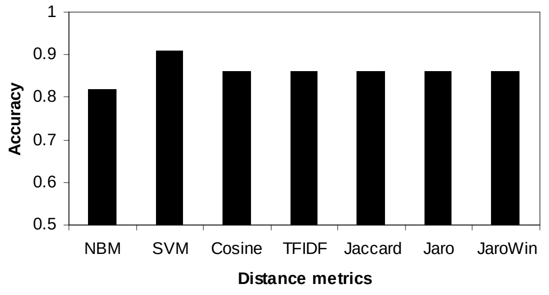
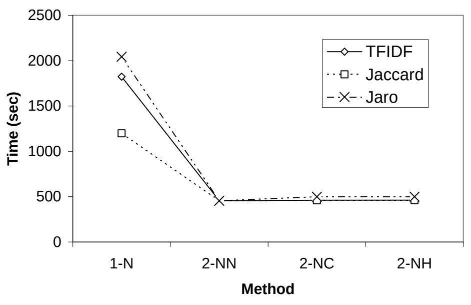


Figure 7: Accuracy (DBLP).

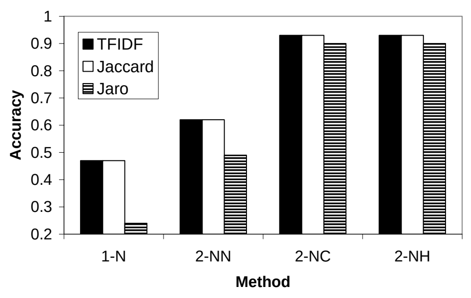
Note that the way we generate artificial name variants may affect the performance of sampling. In general, it is difficult to precisely capture the right percentages of different error types in author name variants. For the original name "Ji-Woo K. Li", for instance, some of possible error types are 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 name disambiguation algorithms, 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%). The accuracy of the former case is shown in Figure 7, and that of the latter case is in Figure 10(a). Note that regardless of the error types or their percentages, both cases show reasonably similar accuracies for all seven distance metrics (i.e., 0.8–0.9 accuracy). Therefore, all subsequent experimentations are done using the latter case (85%/15%).

Evaluation metrics. To measure how effectively name variants can be found, we measured the "accuracy" of top-k as follows. For a name in X, our algorithm finds top-k candidate name variants in Y. If the top-k candidates indeed contain the solution, then it's *match*. Otherwise, it is a *mismatch*. This is repeated for all 100 names in X. Then, overall accuracy is defined as: Accuracy $= \frac{\text{\# of matches}}{100}$. The accuracy was measured for different k values (i.e., $k = 1, 5, 10$). For instance, with $k = 5$ in the DBLP data set, for each author in X, methods return the top-5 candidate name variants out of 364,377 authors, and if one of these 5 candidates is the artificial name variant that we created, then it is a match. We repeated all of the subsequent experiments for three window sizes of 1, 5, and 10, and found that accuracies with larger window size ($k = 10$) are about only 10% higher than those with smaller window size ($k = 1$). Since the different is small, in what follows, we present the results using $k = 5$.

Results. Figure 8 summarizes the experimental results of four alternatives using three representative metrics – TFIDF, Jaccard, and Jaro. In terms of the processing time, 1-N is the slowest for TFIDF 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 that



(a) Scalability



(b) Accuracy

Figure 8: DBLP with $k = 1$.

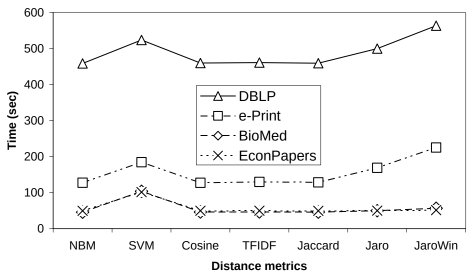


Figure 9: Processing time for Step 2.

exploiting additional information (i.e., coauthor) than the simple name spelling is 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 Figure 9. In general, token-based distance metrics (e.g., TFIDF, 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 10 summarizes the accuracies of our proposal for all four data sets (with $k = 5$). In general, the distance metrics such as the SVM, cosine, TFIDF 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

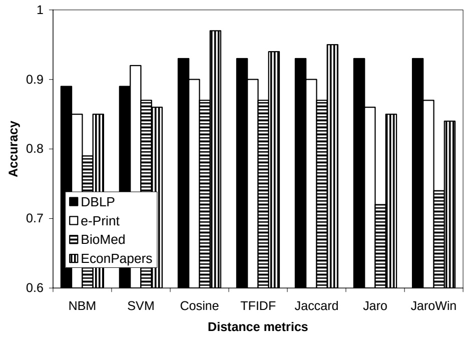


Figure 10: Accuracy ($k = 5$).

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.

In general, Jaro or JaroWinkler method in step 2 gave poorer accuracy than the others. Since they are edit-distance based methods that are heavily affected by the number of transpositions, as the length of string to compare increases (in 2-NC, it is a long coauthor string to compare), its error rate increases as well. In the e-Print data set, the accuracies are lower, compared to those of DBLP. This is because most of citations in e-Print data set use abbreviation for the first name of authors. Since the sampling technique uses TFIDF for weighting tokens, common tokens like abbreviated first name (e.g., "E." or "P.") would have lower weight via IDF, negatively affecting matching process.

6. CONCLUSION

Two interesting and practical problems – *Mixed Citation* and *Split Citation* – are formally introduced and their solutions are explored. Since both problems commonly occur in many of the existing bibliographic digital libraries, it is important to devise effective and efficient solutions to them. By utilizing one of the state-of-the-art sampling-based approximate join techniques, our solutions are scalable yet highly effective. Furthermore, our proposals exploit associated information of author names (e.g., coauthors, titles, or venues) than names themselves, achieving 90-93% accuracy overall.

As to future direction, in addition to comparing ours against others (e.g., [2, 22]), we plan to apply our framework to other domains (e.g., address, movie) to test its generality.

7. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. "Eliminating Fuzzy Duplicates in Data Warehouses". In *VLDB*, 2002.

[2] I. Bhattacharya and L. Getoor. "Iterative Record Linkage for Cleaning and Integration". In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2004.

[3] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. "Adaptive Name-Matching in Information Integration". *IEEE Intelligent System*, 18(5):16–23, 2003.

[4] V. R. Borkar, K. Deshmukh, and S. Sarawagi. "Automatic Segmentation of Text into Structured Records". In *ACM SIGMOD*, 2001.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. "Robust and Efficient Fuzzy Match for Online Data Cleaning". In *ACM SIGMOD*, 2003.

[6] W. Cohen, P. Ravikumar, and S. Fienberg. "A Comparison of String Distance Metrics for Name-matching tasks". In *IIWeb Workshop held in conjunction with IJCAI*, 2003.

[7] N. Cristianini and J. Shawe-Taylor. *"An Introduction to Support Vector Machines"*. Cambridge U. Press, 2000.

[8] J. M. B. Cruz, N. J. R. Klink, and T. Krichel. "Personal Data in a Large Digital Library". In *ECDL*, 2000.

[9] P. T. Davis, D. K. Elson, and J. L. Klavans. "Methods for Precise Named Entity Matching in Digital Collection". In *ACM/IEEE JCDL*, 2003.

[10] I. P. Fellegi and A. B. Sunter. "A Theory for Record Linkage". *J. of the American Statistical Society*, 64:1183–1210, 1969.

[11] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins in an RDBMS for Web Data Integration". In *WWW*, 2003.

[12] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. "Two Supervised Learning Approaches for Name Disambiguation in Author Citations". In *ACM/IEEE JCDL*, Jun. 2004.

[13] M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.

[14] Y. Hong, B.-W. On, and D. Lee. "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach". In *ECDL*, 2004.

[15] J. A. Hylton. *"Identifying and Merging Related Bibliographic Records"*. PhD thesis, Dept. of EECS, MIT, 1996. LCS Technical Report MIT/LCS/TR-678.

[16] M. A. Jaro. "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida". *J. of the American Statistical Association*, 84(406), 1989.

[17] R. P. Kelley. "Blocking Considerations for Record Linkage Under Conditions of Uncertainty". In *Proc. of Social Statistics Section*, pages 602–605, 1984.

[18] S. Lawrence, C. L. Giles, and K. Bollacker. "Digital Libraries and Autonomous Citation Indexing". *IEEE Computer*, 32(6):67–71, 1999.

[19] A. McCallum et al. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". In *ACM KDD*, 2000.

[20] B.-W. On, D. Lee, J. Kang, and P. Mitra. "Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework". In *ACM/IEEE JCDL*, 2005.

[21] H. Pasula et al. "Identity Uncertainty and Citation Matching". In *Advances in Neural Information Processing Systems.* MIT Press, 2003.

[22] S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". In *ACM KDD*, 2002.

[23] M. M. M. Synman and M. Rensburg. "Revolutionizing Name Authority Control". In *ACM DL*, 2000.

[24] J. W. Warnner and E. W. Brown. "Automated Name Authority Control". In *ACM/IEEE JCDL*, 2001.

[25] W. E. Winkler and Y. Thibaudeau. "An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census". Technical report, US Bureau of the Census, 1991.