Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework

Byung-Won On
Dept. of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
on@cse.psu.edu

Dongwon Lee
School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
dongwon@psu.edu

Jaewoo Kang
Dept. of Computer Science
North Carolina State University
Raleigh, NC 27695
kang@csc.ncsu.edu

Prasenjit Mitra
School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
pmitra@ist.psu.edu

ABSTRACT

In this paper, we consider the problem of ambiguous author names in bibliographic citations, and comparatively study alternative approaches to identify and correct such name variants (e.g., "Vannevar Bush" and "V. Vush"). Our study is based on a scalable two-step framework, where step 1 is to substantially reduce the number of candidates via blocking, and step 2 is to measure the distance of two names via coauthor information. Combining four blocking methods and seven distance measures on four data sets, we present extensive experimental results, and identify combinations that are scalable and effective to disambiguate author names in citations.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

Name Disambiguation, Blocking, Measuring Distances

1. INTRODUCTION

Bibliographic Digital Libraries (DLs), such as DBLP [17], CiteSeer [18] or e-Print arXiv [2], contain a large number of citation 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. It is thus an important task to keep the citation records consistent and up-to-date.

However, because of various reasons (e.g., data-entry errors, ambiguous formats, imperfect citation gathering softwares), keeping citations correct and up-to-date proved to be a challenging task in a large-scale DL. While many problems exist, in this work, we specifically focus on the problem of *ambiguous author names*. That is, citations of the same scholar appear under different *name variants*[1] due to errors. Because of this problem, it is difficult to get the complete list of the publications of some authors.

For instance, imagine a scholar "John Doe" has published 100 articles. However, a DL keeps two separate purported author names, "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 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 the following motivational example.

Example 1 (Motivation). In order to demonstrate the needs for such name disambiguation algorithms, we show two real cases drawn from existing digital libraries. Figure 1 is a screen shot of the ACM Portal, which contains the list of author names who have ever published an article in ACM-affiliated conferences or journals. In particular, Figure 1 shows names whose last names are either "Ullman" or "Ullmann". Note that the name of the renowned computer scientist, "Jeffrey D. Ullman" at Stanford university, appears as several variants, *incorrectly*. For instance, "J. D. Ullman" in the list is in fact the same person as "Jeffrey D. Ullman", however they are treated as different scholars.

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

JCDL'05, June 7–11, 2005, Denver, Colorado, USA.

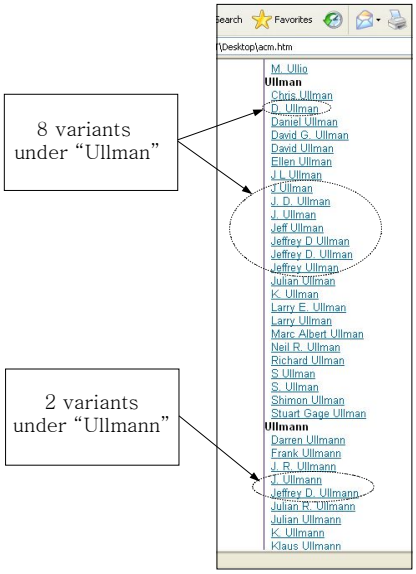


Figure 1: Author name index in ACM Portal.

The second case is drawn from CiteSeer, where we tried to locate all citations about a book, "Artificial Intelligence: A Modern Approach", by "S. Russell" and "P. Norvig". As illustrated in Figure 2, currently, CiteSeer returns 23 different formats of citations of the same book, *incorrectly* thinking that they are all different. Part of the problem is caused by the ambiguous names of the authors. □

As illustrated in Example 1, the problem of ambiguous names prevails in the current DLs. Therefore, locating all name variants of the same author and consolidating their citations into a single definitive name improves the quality of DLs substantially. However, this name disambiguation problem has been exacerbated as the number and volume of DLs increase, posing significant challenges to the management of large-scale DLs. One of the key challenges is to make the name disambiguation algorithms *scalable* and *resilient* in order to cope with the rapid growth of the DLs while not degrading their accuracy. The traditional approaches that rely on the syntactic similarity of name variants (e.g., measuring string edit distance between two names) will likely fail to scale because, as the size of DLs increases, more numbers of common or similar names will appear in DLs, making it increasingly difficult to distinguish them using only their names.

To investigate these problems, in this paper, we introduce a two-step framework, where many alternative methods can be used in each step in order to optimize the performance. In addition, we exploit additional information of coauthor relations in order to further improve the algorithm's accuracy. We identify the contributions of our work as follows.

1. If all author names have to be compared against each other in order to find similar names, the resulting algorithm will be quadratic to the total number of author names in the input. Clearly, this algorithm will not scale for large DLs. In order to address this problem, we advocate a scalable two-step name disambiguation framework. In the first step, the algorithm partitions all author-name strings into a set of disjoint blocks. In the second step, the algorithm visits each block and compare all possible pairs of names within the block in order to find the name variants. This blocking scheme has been introduced and successfully applied in other problem context such as record linkage [8] or identity uncertainty problems [22]. Through an extensive experimental validation, we show the blocking-based pruning methods are very effective for our problem.

2. We consider several alternatives in each step of the framework (e.g., use sampling in the first step and TFIDF distance in the second step), and empirically evaluate the effects of various combinations of such algorithms for various data sets. We conduct an extensive experimental study to validate our claims – four alternatives in the first step and seven alternatives in the second step are examined on four different data domains, a total of $4 \times 7 \times 4 = 112$ combinations.

3. We examine a set of "unsupervised" distance measures, in addition to the two "supervised" algorithms proposed by [11]. Using citations from large real-world DLs, we show that, some of the unsupervised algorithms (e.g., cosine, TFIDF) show comparable or even superior accuracy to supervised ones. Since unsupervised algorithms in general do not require large training sets, they are useful for disambiguating two names with a small number of coauthors. Furthermore, we analyze what kind of combinations in steps 1 and 2 result in good scalability/accuracy trade-offs.

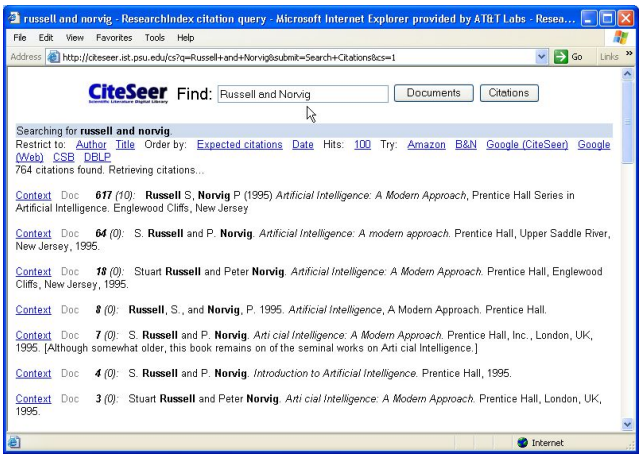


Figure 2: Search result in CiteSeer.

2. RELATED WORK

Han et al. [11] proposed two supervised learning-based approaches for a related but different problem. Their algorithm solves the so called the *citation labeling* problem – given a citation, cross out an author name, and using the remaining citation information, recover the author name via supervised learning methods. We, on the other hand, solve the *name disambiguation* problem – given a collection of author names with different spellings, identify name variants for the same person in reality. In this paper, we adopt their idea of supervised learning methods in the second step of our

framework. The goal of our study is not to compare supervised methods against unsupervised ones. Instead, we are interested in finding combinations of alternatives that give good scalability/accuracy trade-offs in the two-step framework.

Our problem has similarities with a more general class of problems, known as various names – record linkage (e.g., [8, 3]), citation matching (e.g., [22, 20]), identity uncertainty (e.g., [22]), merge-purge (e.g., [12]), object matching (e.g., [5, 25]), duplicate detection (e.g., [23, 1, 21]), approximate string join (e.g., [10]) etc. String similarity measures used in our work were proposed by Jaro [14] and Winkler [26]. 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 Jaro-Winkler, for the name matching task [6]. In DLs, this problem is called citation matching. In the citation matching domain, [16] 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 our 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. [15] in the record linkage literature. Our blocking scheme is also similar in flavor to the two-step citation matching schemes proposed in [13, 20] where initial rough but fast clustering (or called "Canopy") is followed by more exhaustive citation matching step.

3. PROBLEM & SOLUTION OVERVIEW

Problem Definition. We formally define the *name disambiguation* problem as follows:

> Given two *long* lists of author names, X and Y, for each author name x ($\in X$), find a set of author names, y_1, y_2, ..., y_n ($\in Y$) such that both x and y_i ($1 \leq i \leq n$) are name variants of the same author.

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;

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 that are not dependent on the syntactic similarities of author-name strings.

Solution Overview. Figure 3 illustrates our two-step name disambiguation framework. We use the following ideas to design our algorithm: (1) Instead of comparing author-name spellings along to find author-name strings that refer to the same author, we use information associated with the author-name strings like coauthor list, authors' paper titles, venue list that authors often publish, or even institute etc. 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 associated information of an author. Exploiting other associated information (or even hybrid of them as in [11]) is an interesting direction for future work (the case of exploiting both author name and its coauthor is discussed in Section 6.2); (2) To make the algorithm scalable, we borrow the *Blocking* technique popular in the solutions of record linkage problem, and modify the baseline approach as follows:

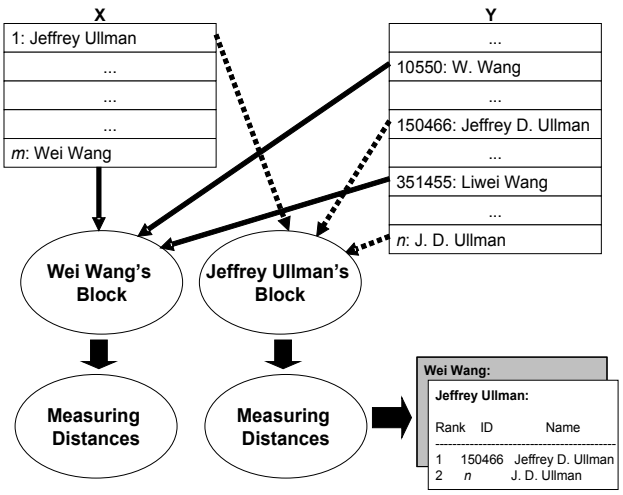


Figure 3: Overview of two-step framework.

/* let C_a be coauthor information of author a; */
for each name x ($\in X$), create a block B_x($\in B$);
for each name y ($\in Y$) /* Step 1 */
  assign y to all relevant blocks B_i($\in B$);
for each block B_x ($\in B$) /* Step 2 */
  for each name z ($\in B_x$)
    if $dist(C_x, C_z) > \phi$, x and z are name variants;

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

4. STEP 1: BLOCKING

The goal of step 1 is to put similar inputs into the same group by some criteria (thus called *Blocking*) so that distance measures of step 2 can be done per group. If the filtering criteria (i.e., blocking methods) are too aggressive, then only small number of inputs will pass them and be put into the same block, and sometimes even right answers may be incorrectly pruned away. On the other hand, if the criteria are too lenient, then too many number of inputs (including noisy data) will pass them, bloating up the size of blocks. Furthermore, in general, the size of blocks has a close correlation with the scalability of the next step – the bigger a block is, the longer it takes to do step 2. In order to see the effects of different blocking schemes in the name disambiguation context, therefore, we examine four representative (and distinct) blocking methods – heuristic, token-based, n-gram, and sampling. Informally, given a set of p author names, n_1, ..., n_p, blocking methods return a set

of q blocks, B_1, ..., B_q as follows:

$$\{B_1, ..., B_q\} \leftarrow \textbf{Blocking}(\{n_1, ..., n_p\})$$

where each B_i contains a set of author names n_j. Note that depending on the blocking scheme, the same author name can be put into multiple blocks.

4.1 Spelling-based Heuristics

The simplest approach is to group author names based on their name spellings, and can be attributed to a more general method known as *sorted neighborhood* [12]. In our context, all names with the same heuristics are grouped into the same block. For instance, "Jeffrey Ullman" is grouped together with "J. Ullman" if the heuristics is "the same initial of the first name and the same last name (iFfL)". Other plausible heuristics are: the same initial of the first name and the same initial of the last name (iFiL), or the same last name (fL), or even the combination of the above. Different heuristics would have slightly different impact on the performance of the two-step methods. For instance, fL would generate bigger sized blocks than iFfL does so that it usually has a higher accuracy while being slower. Since comparing different heuristics is not the goal of this research, and iFfL is the most commonly used in citation matching context [13], in our experimentation, only iFfL is used.

4.2 Token-based

In the token-based blocking, author names sharing at least one common token are grouped into the same block (e.g., "Jeffrey D. Ullman" and "Ullman, Jason"). When authors have rare name spellings, this scheme tends to generate a small sized block. However, when authors' name spellings are common (e.g., "D. Lee"), have long names (e.g., "Juan David Gonzalez Cobas El-Nasr"), or have several initials in the name (e.g., "V. S. P. Srivastava"), then the resulting block can have a large number of names.

4.3 N-gram

The idea of N-gram blocking is similar to that of token-based one, except that it has finer granularity – instead of checking common tokens, this method checks the existence of common N continuous characters from author names (we use $N = 4$ that gave good results in [3]). Since the granularity is finer, the number of author names put into the same block is the largest among the four blocking methods. For instance, using N-gram blocking, "David R. Johnson" and "F. Barr-David" are grouped into the same block because of the common 4-gram "davi".

4.4 Sampling

Another alternative blocking method is to use *Sampling*. Given a name string x, suppose we want to draw a number of samples that are most similar to x, and group them into the same block. If the sampling process is somehow fast while it generates accurate samples, then it can serve as a good blocking method. 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 [10]. We adopt it to our context as follows: Imagine each token from all author names has an associated weight using the TFIDF metric in IR (i.e., common tokens in author names have lower weights while rare ones have higher weights). Then, each author name t is associated with its token weight vector v_t. Suppose that for each name t_q in an author name set R_1, we want to draw a sample of size S from another author name set R_2 such that the frequency C_i of name $t_i \in R_2$ can be used to approximate $sim(v_{t_p}, v_{t_i}) = \sigma_i$. 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 block only if $\frac{C_i}{S} T_V(t_q) \geq \theta$, where θ is a pre-determined threshold[2]. This strategy assures that all pairs of names with similarity of at least θ can be put into the same block with a desired probability, as long as the proper sample size S is given.

5. STEP 2: MEASURING DISTANCES

After a set of blocks are created in step 1, the goal of step 2 is, for each block, to identify top-k author names that are the closest to the name in question. For this, intuitively, we can use various methods that people have developed to measure the distance or similarity of two strings. For this, we have compared two categories of methods – *supervised* and *unsupervised* methods.

5.1 Supervised Methods

Han et al. [11] recently showed the effectiveness of two supervised methods when there are enough number of training data, *naive bayes model* [19] and *support vector machines* [7], in a slightly different name disambiguation context.

5.1.1 The 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." by the probability values of the coauthors of "Dongwon Lee" in testing. As shown in Figure 3, given a block in which there are an author name x of X and authors y_i of Y ($i \in [1, k]$, where k is the total number of authors of Y), we calculate the probability of each pair of x and y_i and find the pair with the maximal posterior probability as follows:

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

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

- $P(Alone|x)$ is the probability of x writing a paper alone.
- $P(Coauthor|x)$ is the probability of x working for a paper with coauthors.
- $P(Frequent|Coauthor, x)$ is the probability of x writing a paper with the coauthors, each of who worked

[2]In experimentation, we used the more optimized version of the sampling-based join approximation with a single scan from [10].

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 1: Terms.

with x at least twice in the training data, conditioned on the event of x's past coauthors.

- $P(A_j|Frequent, Coauthor, x)$ is the probability of x working for a paper with a particular coauthor A_j.

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 is the total number of authors of Y in the block, and k is the k-th coauthor in y_i, who also appears in the coauthor list of x in the training data set.

5.1.2 The Support Vector Machines (SVM)

The *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, 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. 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, or sigmoid kernel).

5.2 Unsupervised Methods

The second group of methods that we consider is the unsupervised ones that do not require any training.

5.2.1 String-based Distance

In this scheme, the distance between two author names are measured by the "distance" between their coauthor lists. That is, to measure the distance between "Dongwon Lee" and "Lee, D.", instead of computing the *dist*("Dongwon Lee", "Lee, D."), we compute the *dist*(coauthor-list("Dongwon Lee"), coauthor-list("Lee, D.")). Among many possible distance measures, we used two token-based string distances (e.g., *Jaccard* and *TFIDF*) and two edit-distance-based ones (e.g., *Jaro* and *Jaro-Winkler*) 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 1, the four metrics can be defined as follows:

- $\mathbf{Jaccard}(\mathbf{x}, \mathbf{y}) = \frac{|T_x \bigcap T_y|}{|T_x \bigcup T_y|}$
- $\mathbf{TFIDF}(\mathbf{x}, \mathbf{y}) = \sum_{w \in T_x \cap T_y} V(w, T_x) \times V(w, T_y)$, 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.
- $\mathbf{Jaro}(\mathbf{x}, \mathbf{y}) = \frac{1}{3} \times (\frac{|CC_{x,y}|}{|C_x|} + \frac{|CC_{y,x}|}{|C_y|} + \frac{|CC_{x,y}| - X_{CC_{x,y}, CC_{y,x}}}{2|CC_{x,y}|})$
- $\mathbf{Jaro - Winkler}(\mathbf{x}, \mathbf{y}) = Jaro(x, y) + \frac{max(|L|,4)}{10} \times (1 - Jaro(x, y))$, where L is the longest common prefix of x and y.

5.2.2 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 in the block. For example, suppose we have a block that has three groups of citations, one for "D. Lee", another for "D. W. Lee" and the other for "D. Y. Lee". In the first group, suppose we have five papers, each coauthored by "D. Lee" with one or more of "J. Kang", "P. Mitra", and "T. Han". Further suppose among the five papers "D. Lee" coauthored three times with "J. Kang", four times with "P. Mitra", and once with "T. Han". Similarly, suppose we have 10 papers in the second group, in which "D. W. Lee" coauthored five times with "J. Kang", seven times with "P. Mitra", and three times with "W. Chu". In the last group, suppose we have seven papers in which "D. Y. Lee" coauthored three times with "P. Chopra", five times with "J. Xu", and once with "J. Kang".

The total number of unique coauthor names in the block is 6 except the three "D. Lee" name variants. In order to determine if the three name variants are indeed referring to the same person, we model the coauthor information for each variant as a vector and compute the distances between the vectors. The resulting vectors have 12 dimensions, each of which corresponds to one of 6 unique coauthor names appearing in the block. For example, for the first group of citations for "D. Lee", we have a vector v("D. Lee") = [0 0 1 3 4 0], provided that the dimensions are ordered by coauthors' last name and values in the vector represent the number of papers that the author represented by the dimension coauthored with "D. Lee". For example, the value, 1, in the third dimension represents the number of papers that "T. Han" coauthored with "D. Lee". Similarly, we have v("D. W. Lee") = [0 3 0 5 7 0] and v("D. Y. Lee") = [3 0 0 1 0 5]. In order to measure the distance between the vectors, v and w, we use the simple cosine distance, an angle between two vectors, defined as: $\cos\theta = \frac{v \cdot w}{\|v\| \cdot \|w\|}$.

5.3 Examples

Table 2 shows the contents of a block with three names after step 1 – "Jeffrey D. Ullman", "J. D. Ullman", and "Daniel Ullman". Using this data, let us compute two distances: **D1** = *dist*("Jeffrey D. Ullman", "J. D. Ullman") and **D2** = *dist*("Jeffrey D. Ullman", "Daniel Ullman") for the naive bayes model and the support vector machines.

First, consider how to compute both distances using the NBM. "Jeffrey D. Ullman" has four coauthors – "Alfred V. Aho", "John E. Hopcroft", "Fereidoon Sadri", and "David Maier". Suppose the papers with ID 1 and 3 are chosen as training data. Then, there are 3 coauthors in the training set. Let us call it as X. Next, we need to calculate the probabilities of coauthors conditioned on the event of

Name	ID	Coauthors
"Jeffrey D. Ullman"	1	"Jeffrey D. Ullman"
	2	"Jeffrey D. Ullman", "Alfred V. Aho", "John E. Hopcroft"
	3	"Fereidoon Sadri", "Jeffrey D. Ullman", "Alfred V. Aho", "David Maier"
"J. D. Ullman"	4	"David Maier", "J. D. Ullman"
	5	"Rajeev Motwani", "Alfred V. Aho", "Fereidoon Sadri", "J. D. Ullman"
	6	"Sergey Brin", "Alfred V. Aho", "J. D. Ullman", "David Maier"
"Daniel Ullman"	7	"Walter Stromquist", "Daniel Ullman"
	8	"James Gary Propp", "Robin Pemantle", "Aviezri S. Fraenkel", "Daniel Ullman"

Table 2: Contents of an example block.

Distance	Cosine	TFIDF	Jaccard	Jaro	JaroWinker
D1	0.89	0.74	0.59	0.75	0.78
D2	0.0	0.08	0.04	0.67	0.71

Table 3: Distance results of unsupervised methods.

"Jeffrey D. Ullman" using the Bayes rule as follows: $P(1) = P(\text{"Fereidoon Sadri"}|\text{"Jeffrey D. Ullman"}) = 0.5$, $P(2) = P(\text{"Alfred V. Aho"}|\text{"Jeffrey D. Ullman"}) = 0.5$, and $P(3) = P(\text{"David Maier"}|\text{"Jeffrey D. Ullman"}) = 0.5$. In testing, we try to find an author name with a maximal posterior probability. For instance, the author name "J. D. Ullman" shares three coauthors with "Jeffrey D. Ullman" – "Fereidoon Sadri", "Alfred V. Aho", and "David Maier" (i.e., $|Y| = 3$). Then, using the formula in Section 5.1.1, the probability of "J. D. Ullman" (i.e., D1) is computed as: $\frac{|Y|}{|X|} \times P(1) \times P(2) \times P(3) = 1.0 \times 0.5 \times 0.5 \times 0.5 = 0.125$. The probability of "Daniel Ullman" can be computed similarly and is 0.0. Two distances, D1 and D2, for the five unsupervised methods are shown in Table 3.

6. EXPERIMENTAL COMPARISON

6.1 Set-up

Data sets. We have gathered real citation data from four different domains, as summarized in Table 4. 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 citations also vary. For instance, citations in e-Print use the first name of authors as only initial, while ones in DBLP use full names.

Artificial name variants. Ideally, it would be desirable to apply our framework to existing DLs to find all real name variants. However, given the 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" and "B" are indeed name variants, human experts had to trace it carefully. Therefore, we use artificial 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 from hundreds to thousands.

To make solution sets, for each data set, we prepare two lists of author names, X and Y, where X is initially empty, and Y contains the entire author names (e.g., 364,377 names for DBLP). Then, we pick top-100 author names from Y according to their number of citations, and generate 100 corresponding new name variants artificially. Note that the reason that we have to use "top 100" instead of "random 100" is because of the two supervised methods in step 2. That is, the two supervised methods, first proposed in [11], do not work without enough numbers of training sets. 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 splitting the original 344 citations into halves, each name carries half of citations, 172, and is put back into X and Y, respectively. At the end, there are 100 and 364,377 names in X and Y, respectively. Then, through the proposed two-step name disambiguation framework, 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 blocking. For instance, if all artificial names that we generated are abbreviation of first names, then both heuristic and 4-gram blocking would work well. However, if artificial name variants were generated by adding garbage characters in last names, then this will negatively affect the performance of heuristic blocking, while it has little effect on 4-gram blocking method. 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", 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 4, and that of the latter case is in Figure 9(a). Note that regardless of the error types or their percentages, all blocking methods, except iFfL, show reasonably similar accuracies. That is, since both token-based and sampling-based blocking have a granularity of "tokens" in processing, they are more tolerable than iFfL which can only handle well name abbreviation error type. Likewise, since the granularity of 4-gram is the finest (i.e., characters), it shows the best tolerance for diverse error types. All subsequent experimentations are done using the latter case (85%/15%).

Implementations. We have implemented various algorithms of Sections 4 and 5 as follows: (1) *Step 1*: Token-

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 4: Summary of data sets.

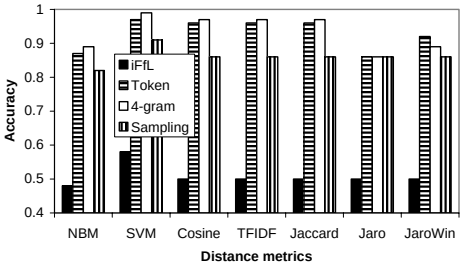


Figure 4: Accuracy with various error types (DBLP).

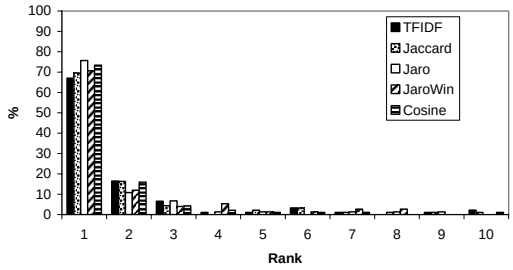


Figure 5: Distribution of name variants in top-10.

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 5: Solution space.

based and N-gram blocking methods were implemented by the open source tool, SecondString [24], where all space-delimited words from coauthor lists are treated as tokens and "4" was used for N-gram (i.e., 4 continuous characters) as tokens. For the sampling-based blocking, we used the implementation of [10] with a sample size of 64 and a threshold of 0.1, and ran the experimentation in Microsoft SQL Server 2000. (2) *Step 2*: For the supervised learning methods, citations per author were randomly split, with half of them used for training, and the other half for testing. For the implementation of Support Vector Machines, LIBSVM [9] was used. In particular, we found that the multi-classifier based implementation in [11] performs poorly for our experimentation. This is because a classifier needs to be prepared for each candidate, and there are usually large number of candidates in our large-scale setting. Therefore, instead, our implementation of SVM is binary-classifier based. In the comparison, the binary-classifier based SVM showed an accuracy about 25% higher than multi-classifier based one (while taking about 20% less time). For the string-based distance functions of the unsupervised learning methods, we used the implementations of TFIDF, Jaccard, Jaro, and Jaro-Winkler from SecondString [24].

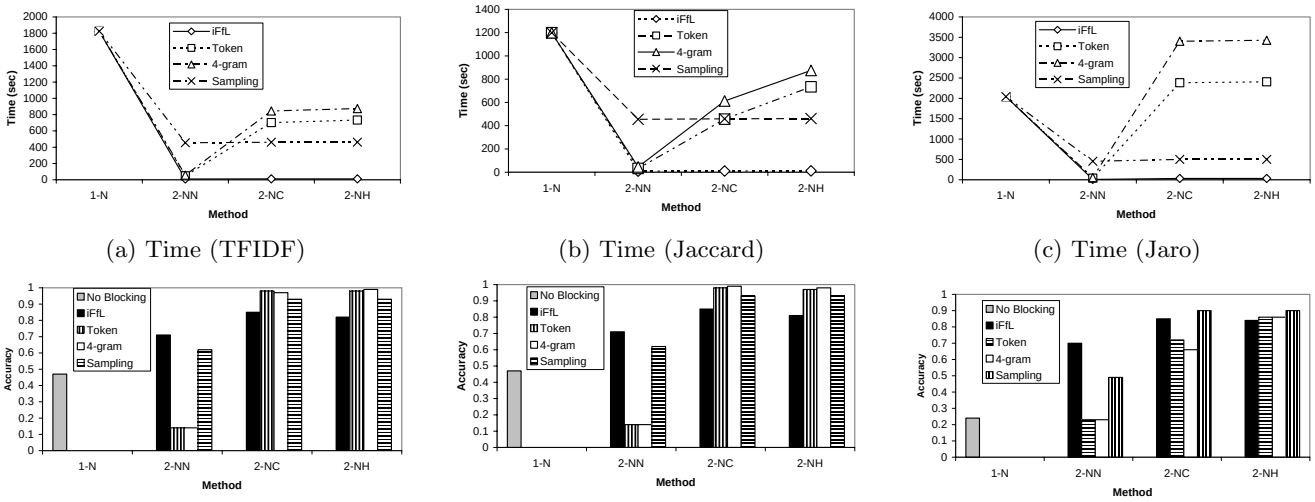
Evaluation metrics. Two main metrics that we used are *scalability* and *accuracy*: (1) The scalability of the framework was measured by the "size" of blocks generated in step 1 and the "time" it took to process both steps 1 and 2; (2) To measure how effectively name variants can be found, we measured 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 is a *match*. Otherwise, it is a *mis-match*. 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 10% higher than those with smaller window size ($k = 1$). The seemingly only-small drop of the accuracy can be explained as follows. As shown in Figure 5, about 70% of correct name variants are returned as the first among 10 candidates. That is, even if we use a smaller window size ($k = 1$), 70% of name variants would be found correctly. Since the most of name variants are found within rank 3, when we used $k = 5$, its accuracy is almost as good as that of $k = 10$. In the following, therefore, we show the results for $k = 5$.

6.2 Our Framework vs. Other Alternatives

Table 5 summarizes four possible approaches to the given problem: (1) 1-N is a single-step pair-wise name matching scheme without using blocking and coauthor information; (2) 2-NN uses the two-step approach, but do not exploit coauthor information; (3) 2-NC is the main proposal in our framework; 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).

Figure 6 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). However, Figure 6(c) shows a rather unexpected result in that both 2-NC and 2-NH take



(a) Time (TFIDF) (b) Time (Jaccard) (c) Time (Jaro)

(d) Accuracy (TFIDF) (e) Accuracy (Jaccard) (f) Accuracy (Jaro)

Figure 6: Comparison of four alternatives (DBLP with $k = 1$).

more time than 1-N does, despite their advantages through blocking in step 1. This is due to the combination of the slow distance computation of Jaro method and long coauthor names, and can be explained as follows. To make the discussion simple, suppose computing $dist(x, y)$ takes 1 sec, and its computation time is proportional to the number of characters in x and y. That is, $dist(x, y)$ takes 1 sec, while $dist(x^{100}, y^{20})$ takes $100 \times 20 = 2,000$ sec (if x^n denotes characters x with the length of n).

Then, (1) Since 1-N computes $dist(x, y)$ for all pairs from X and Y, it takes $|X| \times |Y| = 100 \times 364,377 \approx 36.4M$ sec. (2) In Section 6.1, we chose top-100 authors with the most number of citations as our targets. Since these 100 authors have a large number of citations, their number of coauthors is large too (i.e., on average 100 coauthors per author). Therefore, using 2-NC, computing $dist(x', y')$ (i.e., x' and y' are coauthors of x and y, respectively) is equal to computing $dist(x^{100}, y^{100})$, taking about $100 \times 100 = 10,000$ sec. Since all names in a block must be compared, if a block has 3,500 names (e.g., 4-gram blocking in Figure 7), then it takes $3,500 \times dist(x', y') = 3,500 \times 10,000 \approx 35M$ sec. Note that the time for 1-N, $36.4M$ sec, is roughly the same as that for 2-NC, $35M$ sec. That is, when computation-heavy distance metrics such as Jaro or Jaro-Winkler are used in step 2, and since coauthor names to consider are very long, the expense offset the benefit of the blocking in step 1. Note that our 100 target names in testing are those with the largest number of coauthors. Therefore, the scenario in our experimentation is the "worst case" for 2-NC.

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 exploiting additional information than the simple name spelling is beneficial in the name disambiguation problem. Compared to 2-NC, 2-NH shows no significant improvement. However, for Jaro method (Figure 6(f)), the accuracy of 2-NH, when token-based or 4-gram blocking is used, improves by 10%-15% from 2-NC. Note that Jaro method tends to work better when an input string is short. Therefore, when both name and coauthor

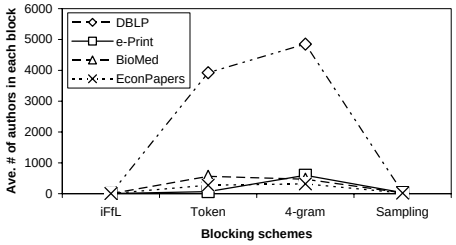


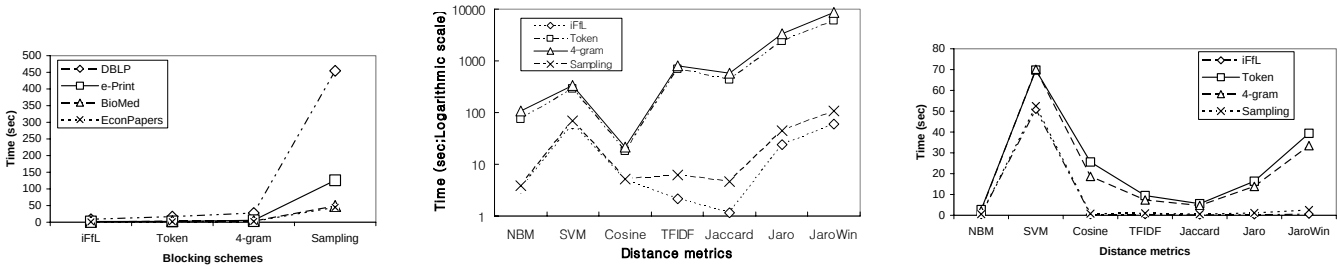
Figure 7: Average # of authors per block.

are considered in 2-NH, Jaro takes advantage of relatively good accuracy from "name" side, although it suffers from "coauthor" side. At the end, this results in about 10%-15% improvements of accuracy. Since 2-NH takes longer than 2-NC while it shows only a negligible improvement in accuracy, in the remaining experiments, we use 2-NC as our default scheme.

6.3 Scalability

Figure 7 illustrates the average # of authors per each block. Regardless of the data set, 4-gram blocking generates the most number of author names into each block. On the other hand, both heuristic and sampling blocking methods put a small number of author names into each block. Nevertheless, the time it takes to perform the blocking is quite small, except the sampling-based blocking which needs a pre-processing for TF/IDF weighting and sampling.

The processing time for step 2 is shown in Figure 8(b)-(c) for DBLP (364,377 authors) and EconPapers (18,399 authors) data sets. The other two data sets have similar graphs to that of EconPapers and omitted. In general, cosine similarity method is the fastest and edit-distance based distance metrics such as Jaro or Jaro-Winkler are the slowest. This is especially so since the string to compare is a long coauthor list (instead of short author name). Both NBM and SVM are relatively faster than TFIDF, Jaccard, Jaro, and



(a) Step 1 (b) Step 2 (DBLP) (c) Step 2 (EconPapers)

Figure 8: Processing time for Steps 1 and 2.

	NBM	SVM	Cosine
iFfL	3.452	59.088	5.057
Token	73.509	292.752	17.625
4-gram	108.362	334.819	21.182
Sampling	3.906	69.092	5.176

Table 6: Processing time for step 2 of NBM, SVM, and Cosine methods.

Jaro-Winkler. Note that those token-based distance metrics such as TFIDF and Jaccard are slower than NBM, SVM, and cosine methods, because there are a large number of candidate name variants in each block. Detailed comparison of cosine vs. two supervised methods (NBM and SVM) are shown in Table 6. Although all three appear to take the same processing time in Figure 8 due to the enlarged Y-axis, cosine method is in fact much faster than the others, showing a better scalability. The SVM takes more time than the others since the hyperplane needs to be split in succession due to SVM's binary-classifiers.

6.4 Accuracy

Figure 9 summarizes the accuracies of four blocking methods of step 1 combined with seven distance metrics of step 2 for all four data sets (with $k = 5$). The case of EconPapers data set is omitted since it is quite similar to that of BioMed data set. Several phenomenons are noticeable.

In general, the distance metrics such as the SVM, cosine, TFIDF and Jaccard perform much better than the others, regardless of the blocking methods used in step 1. For instance, for the DBLP, the four methods achieved near perfect accuracies finding all 100 name variants out of 364,377 candidates. The similar accuracies are observed for e-Print data set as well. Although their accuracies drop to about 0.9 for BioMed, they are still outperforming the other methods such as NVM, Jaro, or Jaro-Winkler. The reason of the lower accuracy for BioMed data set can be explained next. Although fast, the accuracy of spelling-based heuristics such as iFfL is poor throughout all experimentations. This is because it is incapable of handling various error types in name variants (e.g., "J. Ullman" and "Jef. D. Ullmannn").

The accuracies of DBLP and e-Print data sets are better than that of BioMed (and the omitted EconPapers) 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 the 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.

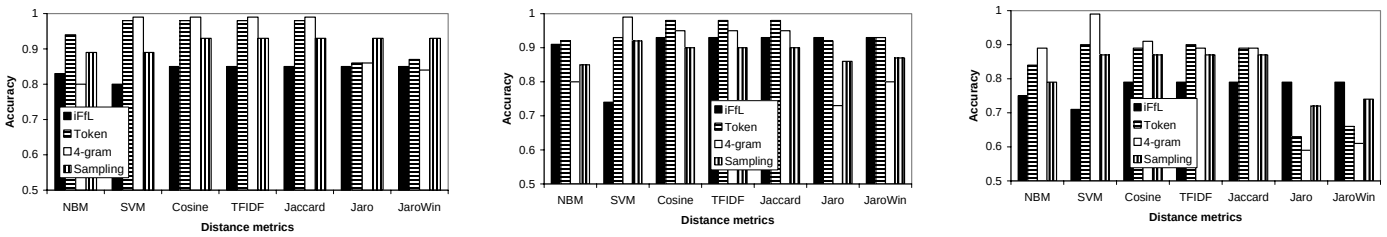
In general, Jaro or Jaro-Winkler method in step 2 gave poorer accuracy than the others. Since they are edit-distance based methods that are heavily affected by the number of transpositions, as the length of string to compare increases (in 2-NC, it is a long coauthor string), its error rate increases as well. In the e-Print data set, the accuracies are lower, compared to those of DBLP, when the sampling-based blocking is used in step 1. This is because most of citations in the e-Print data set use abbreviation for the first name of authors (e.g., "F. E. Bauer" or "E. Fernandez"). Since the sampling technique use TFIDF for weighting tokens, common tokens like abbreviated first name (e.g., "E.") would have lower weight via IDF, negatively affecting matching process. This is why the sampling-based blocking performed worse than the plain token-based blocking.

6.5 Summary of Experiments

In short, 2-NC showed a better scalability and accuracy compared to 1-N or 2-NN, validating our assumption that using associated information than name itself would be beneficial in disambiguating name variants. One could even get a better accuracy with 2-NH at the cost of time. When 2-NC was used, a combination of token-based or N-gram blocking (step 1) and SVM as a supervised method or cosine metric as a unsupervised method (step 2) gave the best scalability/accuracy trade-off. In addition, this combination was tolerable to various error types in names. Finally, the accuracy of simple name spelling based heuristics were shown to be quite sensitive to the error types, while edit distance based distance metrics such as Jaro or Jaro-Winkler proved to be inadequate for large-scale name disambiguation problem for its slow processing time.

7. CONCLUSION

Based on our scalable two-step framework, we compared various configurations – four blocking in step 1 (i.e., heuristic, token, N-gram, sampling), seven distance metrics via



(a) DBLP (b) e-Print (c) BioMed

Figure 9: Accuracy comparison ($k = 5$).

"coauthor" information in step 2 (i.e., NBM, SVM, Cosine, TFIDF, Jacard, Jaro, Jaro-Winkler), against four data sets (i.e., Computer Science, Physics, Medical, Economics). Experimental results verify that our proposed two-step framework using coauthor relation (instead of author name alone) shows much improved scalability and accuracy (e.g., 4 times faster and 50% more accurate using sampling/TFIDF on DBLP data set) compared to one-step approach.

8. REFERENCES

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

[2] arXiv.org e Print archive. http://arxiv.org/.

[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*, Santa Barbara, CA, May 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 University Press, 2000.

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

[9] A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

[10] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins in an RDBMS for Web Data Integration". In *Int'l World Wide Web Conf. (WWW)*, 2003.

[11] H. Han, C. L. Giles, and H. Zha et al. "Two Supervised Learning Approaches for Name Disambiguation in Author Citations". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2004.

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

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

[14] 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), Jun. 1989.

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

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

[17] M. Ley. "The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives". In *Int'l Symp. on String Processing and Information Retrieval (SPIRE)*, Lisbon, Portugal, Sep. 2002.

[18] CiteSeer: Scientific Literature Digital Library. http://www.citeseer.org/.

[19] B. Majoros. "Naive Bayes Models for Classification". http://www.geocities.com/ResearchTriangle/Forum/1203/NaiveBayes.html.

[20] A. McCallum, K. Nigam, and L. H. Ungar. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". In *ACM KDD*, Boston, MA, Aug. 2000.

[21] A. E. Monge. *"Adaptive Detection of Approximately Duplicate Database Records and the Database Integration Approach to Information Discovery"*. PhD thesis, University of California, San Diego, 1997.

[22] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. "Identity Uncertainty and Citation Matching". In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

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

[24] SecondString: Open source Java-based Package of Approximate String-Matching. http://secondstring.sourceforge.net/.

[25] S. Tejada, C. A. Knoblock, and S. Minton. "Learning Object Identification Rules for Information Integration". *Information Systems*, 26(8):607–633, 2001.

[26] 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.