Effective Self-Training Author Name Disambiguation in Scholarly Digital Libraries

Anderson A. Ferreira   Adriano Veloso   Marcos André Gonçalves   Alberto H. F. Laender

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
31270-901 Belo Horizonte, Brazil
{ferreira, adrianov, mgoncalv, laender}@dcc.ufmg.br

ABSTRACT

Name ambiguity in the context of bibliographic citation records is a hard problem that affects the quality of services and content in digital libraries and similar systems. Supervised methods that exploit training examples in order to distinguish ambiguous author names are among the most effective solutions for the problem, but they require skilled human annotators in a laborious and continuous process of manually labeling citations in order to provide enough training examples. Thus, addressing the issues of (i) automatic acquisition of examples and (ii) highly effective disambiguation even when only few examples are available, are the need of the hour for such systems. In this paper, we propose a novel two-step disambiguation method, SAND (Self-training Associative Name Disambiguator), that deals with these two issues. The first step eliminates the need of any manual labeling effort by automatically acquiring examples using a clustering method that groups citation records based on the similarity among coauthor names. The second step uses a supervised disambiguation method that is able to detect unseen authors not included in any of the given training examples. Experiments conducted with standard public collections, using the minimum set of attributes present in a citation (i.e., author names, work title and publication venue), demonstrated that our proposed method outperforms representative unsupervised disambiguation methods that exploit similarities between citation records and is as effective as, and in some cases superior to, supervised ones, without manually labeling any training example.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Retrieval; I.5.2 [**Pattern Recognition**]: Classifier design and evaluation

General Terms

Algorithms, Experimentation

JCDL'10, June 21–25, 2010, Gold Coast, Queensland, Australia.
Copyright 2010 ACM 978-1-4503-0085-8/10/06 ...$10.00.

Keywords

Name Disambiguation, Bibliographic Citations

1. INTRODUCTION

Several scholarly digital libraries (DLs), such as DBLP[1], CiteSeer[2], MEDLINE[3] and BDBComp[4], provide features and services that facilitate literature research and discovery as well as other types of functionality. Such systems may list millions of bibliographic citation records (here understood as a set of bibliographic features such as author and coauthor names, work title and publication venue title, of a particular publication) and have become an important source of information for academic communities since they allow the search and discovery of relevant publications in a centralized way. Also, studies about the DL content can lead to interesting results such as coverage of topics, tendencies, quality and impact of publications of a specific sub-community or individuals, patterns of collaboration in social networks, etc. These types of analysis and information, which are used, for example, by funding agencies for decisions on grants and for individual's promotions, presuppose *high quality content* [20, 22].

Citation management within DLs involves a number of tasks. One in particular, *author name disambiguation*, has required a lot of attention from the DL research community due to its inherent difficulty. Specifically, name ambiguity is a problem which occurs when a set of citation records contains ambiguous author names (the same author may appear under distinct names, or distinct authors may have similar names). This problem may be caused by a number of reasons, including the lack of standards and common practices, and the decentralized generation of content (e.g., by means of automatic harvesting).

The name disambiguation task may be formulated as follows. Let $C = \{c_1, c_2, ..., c_k\}$ be a set of citation records. For each author in a citation record c_i, an authorship record r_i is created to represent his/her participation in that citation. The objective is to produce a disambiguation function which is used to partition the set of authorship records into n sets $\{a_1, a_2, \ldots, a_n\}$, so that each partition a_i contains (all and ideally only all) the authorship records in which the ith author appears.

[1] http://dblp.uni-trier.de

[2] http://citeseer.ist.psu.edu

[3] http://medline.cos.com/

[4] http://www.lbd.dcc.ufmg.br/bdbcomp

To disambiguate the bibliographic citations of a digital library, first we may split the set of authorship records into groups of ambiguous authors, called ambiguous groups (i.e., groups of citations having authors with similar names). The ambiguous groups may be obtained, for instance, by using a blocking method [27]. Blocking methods address scalability issues, avoiding the need for comparisons among all authorship records.

The challenges of dealing with name ambiguity in bibliographic DLs have led to a myriad of disambiguation methods [4, 5, 8, 9, 14, 15, 16, 17, 18, 19, 23, 25, 26, 27, 28, 33, 34, 35, 41]. However, despite the fact that most of these methods were demonstrated to be relatively effective (in terms of error rate or similar metrics), none of them provides a perfect and final solution for the problem (i.e., they produce errors). Existing disambiguation methods usually follow either an *unsupervised* or a *supervised* approach. In the former case, the methods exploit similarities between authorship records in order to place in the same group those records that belong to the same author. In the latter case, the methods exploit a set of training examples, from which a disambiguation function is derived and then used to place authorship records in the corresponding group.

Supervised methods are usually the most effective ones for name disambiguation. In more details, we are given as input a set of authorship records called the *training data* (denoted as $\mathcal{D}$) that consists of examples or, more specifically, records for which the correct authorship is known. Each example is composed of a set $\mathcal{F}$ of m features $\{f_1, f_2, \ldots, f_m\}$ along with a special variable called the *author*. This *author* variable draws its value from a discrete set of labels $\{a_1, a_2, \ldots, a_n\}$, where each label uniquely identifies an author. The training examples are used to produce a disambiguation function (i.e., the disambiguator) that relates the features in the training examples to the correct author. The *test set* (denoted as $\mathcal{T}$) for the disambiguation task consists of a set of authorship records for which the features are known while the correct author is unknown. The disambiguator, which is a function from $\{f_1, f_2, \ldots, f_m\}$ to $\{a_1, a_2, \ldots, a_n\}$, is used to predict the correct author for the records in the test set. In this context, the disambiguator essentially divides the records in $\mathcal{T}$ into n sets $\{a_1, a_2, \ldots, a_n\}$, where a_i contains (ideally all and only all) the authorship records in which the ith author is included. Alternatively, the disambiguator may take as input a pair of authorship records and outputs a binary decision whether these records belong to same author.

Although successful cases of the application of supervised methods have been reported [9, 14, 17, 36, 35, 37], the acquisition of training examples requires skilled human annotators to manually label authorship records. DLs are very dynamic systems, thus manual labeling of large volumes of examples is unfeasible. Further, the disambiguation task presents nuances that impose the need for methods with specific abilities. For instance, since it is not reasonable to assume that all possible authors are included in the training data, disambiguation methods must be able to detect unseen authors, for whom no label was previously assigned (i.e., there is no authorship records for these authors in the training data).

Unsupervised methods, on the other hand, require no manual labeling effort, since they simply group authorship records into clusters by maximizing intra-cluster similarity while minimizing inter-cluster similarity. Obviously, the choice of a proper similarity measure is of paramount importance, and a natural choice is to employ similarity measures based on highly discriminative features, such as coauthor names. In this case, the resulting clusters are very likely to be pure, in the sense that each cluster is likely to contain only authorship records of the same author. The drawnback, however, is that some authors are likely to have their authorship records fragmented into several (pure) clusters, compromising the effectiveness of unsupervised methods.

In this paper, we propose a hybrid disambiguation method, which will hereafter be referred to as SAND (standing for Self-training Associative Name Disambiguator). SAND exploits the strengths of both unsupervised and supervised methods. Specifically, it works in two steps. In the unsupervised step, recurring patterns in the coauthorship graph are exploited in order to produce pure clusters of authorship records. Then, in the supervised step, a subset of the extracted clusters is provided as training, from which a disambiguation function is derived. The final result is a highly effective and extremely practical disambiguator, as will be shown in a set of experiments using citation records extracted from the DBLP and BDBComp collections. The results show that SAND outperforms unsupervised methods in more than 27% on the DBLP collection and 4% on the BDBComp collection. Improvements when compared against supervised methods are also reported.

The rest of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we describe the proposed hybrid method, SAND. In Section 4 we present the evaluation of SAND and compare its effectiveness with the effectiveness provided by other representative methods. Finally, in Section 5 we conclude paper.

2. RELATED WORK

The name disambiguation methods proposed in the literature adopt a wide spectrum of solutions [32] that include approaches based on manual assignment by librarians [31], collaborative efforts[5], unsupervised techniques [4, 5, 8, 15, 16, 18, 19, 23, 25, 26, 28, 33, 34, 41] and supervised techniques [9, 14, 17, 36, 35, 37].

The unsupervised methods, i.e., methods based on unsupervised techniques, usually exploit similarities between authorship records, by means of predefined similarity/dissimilarity functions, in order to group those records that are likely to be associated with the same author. These functions are defined over existing features in the citations [5, 8, 16, 23, 25, 26, 33], or over implicit information, such as citation topics [34, 41] or data retrieved from the Web [18, 19, 28, 41]. There are also unsupervised methods that attempt to identify the author of the records in an iterative way [4, 15].

Supervised methods may derive a function to predict the author of an authorship record [14] or to inform if two authorship records belong to the same author [9, 17, 36, 35, 37]. In the first case, the records in the training data and test set represent authorship records, while in the last case, the records in the training data and test set represent the comparison between two authorship records.

Since name ambiguity is a practical problem that is not restricted to a single context and comes up in a variety of

[5] http://meta.wikimedia.org/wiki/WikiAuthors

scenarios, it is worth noting that several other name disambiguation methods, that exploit the most diverse types of information (e.g., institutions, addresses, etc.) or are targeted to other applications (e.g., name disambiguation in Web search results), have been described in the literature. For instance, Vu et al. [40] propose the use of Web directories as a knowledge base to disambiguate personal names in Web search results, whereas Bekkerman and McCallum [3] present two methods for addressing this same problem, one based on the link structure of the Web pages and the other one using agglomerative/conglomerative double clustering, a multi-way distributional clustering. Diehl et al. [10] consider the name ambiguity problem in the context of organizational email archives. Galvez and de Moya Anegón [12] address the problem of conflating personal name variants in a canonical form using binary matrices and finite-state graphs. A more detailed discussion of these methods is, however, out of the scope of this paper.

Our proposed method differs from those cited above as it combines properties from both unsupervised and supervised methods. To the best of our knowledge, the closest work is [35], where the examples are automatically extracted using MEDLINE attributes. However, that work is still very different from ours, as the proposed heuristics are based on attributes that were not exploited in this work, such as MeSH terms and author's affiliation.

3. SELF-TRAINING NAME DISAMBIGUATION

In this section we propose SAND, a hybrid disambiguation method that follows a two-step approach. These steps are applied after a well-known pre-processing procedure, which includes blocking, stop-word removal, and stemming[6]. In the following sections we present a detailed description of SAND steps.

3.1 The Unsupervised Step

The objective of this step is to automatically produce training examples. The approach we adopted is to organize authorship records within each ambiguous group into clusters, so that records that are placed in the same cluster tend to be similar to each other while being dissimilar to records placed in other clusters. The key intuition is to associate each cluster with an author label, so that authorship records within each cluster can be exploited as training examples.

In order to work properly as training examples, the extracted clusters must be as pure as possible, in the sense that each cluster must be likely to contain only authorship records that are associated with the same author. Otherwise, if a cluster with low degree of purity is provided as training, then authorship records associated with different authors would receive the same author label, being inappropriate for training (cluster L_3 in Figure 1 (a)).

A simple way to extract pure clusters is to ensure that each cluster contains only one authorship record. In this case, clusters are totally pure, however, totally fragmented (i.e., authorship records associated with the same author are placed in different clusters). Fragmented clusters are detrimental for training, since authorship records associated with the same author would receive different author labels (Figure 1 (b)). Thus, our strategy to automatically produce examples is to extract pure clusters, and then discard those clusters that increase fragmentation (cluster L_3 was discarded in Figure 1 (c)). This strategy is discussed next.

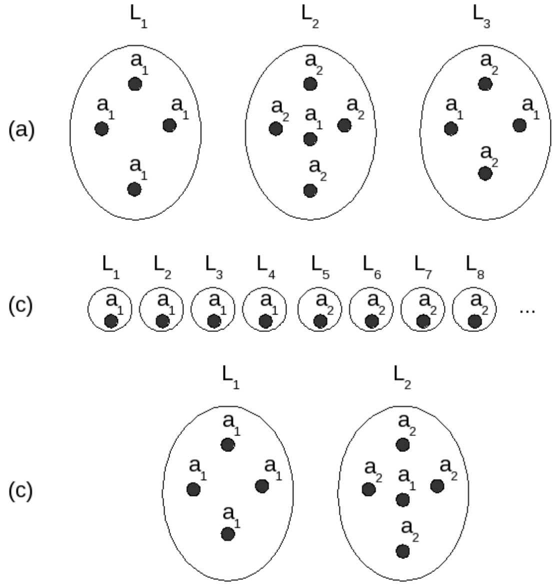

Figure 1: $\mathbf{a}_1$ and $\mathbf{a}_2$ are ambiguous authors. (a) $\mathbf{L}_1$ and $\mathbf{L}_2$ are clusters with high degree of purity. Cluster $\mathbf{L}_3$ is not pure (records associated with author $\mathbf{a}_1$ receive the same label as records associated with author $\mathbf{a}_2$). (b) Clusters are pure, but totally fragmented (records associated with author $\mathbf{a}_1$ receive different labels). (c). Only records in clusters $\mathbf{L}_1$ and $\mathbf{L}_2$ are used as examples.

Extracting Pure Clusters

Pure clusters are extracted by exploiting highly discriminative features, so that records associated with different authors are unlikely to be grouped together in the same cluster. More specifically, pure clusters are extracted by exploiting recurring patterns in the coauthorship graph, that is, two authorship records are placed together in the same cluster if both records have at least one coauthor in common. We have based this strategy on the general observation that only very rarely two ambiguous authors share a coauthor [8]. While simple, this strategy tends to extract highly pure clusters, as it will be shown in our experiments. Unfortunately, this strategy also tends to fragment authorship records associated with the same author into multiple clusters. This is expected, since some authors are likely to have many different coauthors and these coauthors may not act as coauthors among themselves.

Discarding Fragmented Clusters

As mentioned before, if the final set of clusters to be used as training data is fragmented, then possibly many author-

[6] Stop-word removal and stemming are performed on the words that compose work and publication venue titles. Then, authors with ambiguous names are grouped together (i.e., blocked), so that ambiguous groups are created. Disambiguation operations are performed within each ambiguous group, so that useless comparisons involving non-ambiguous authors are avoided.

ship records will be associated with incorrect author labels, hurting the correctness of the training examples. One strategy to reduce fragmentation is to remove those clusters that contribute to increase fragmentation.

The process of identification of fragmented clusters starts by sorting them in descending order of size (i.e., the number of authorship records in the cluster). The result is a sorted list $\mathcal{C}$ of clusters. The identification process continues and, at the first iteration, the largest cluster in $\mathcal{C}$ is inserted into the set of selected clusters, denoted as $\mathcal{S}$. Also, the selected cluster is removed from $\mathcal{C}$. The next clusters in $\mathcal{C}$ to be inserted into $\mathcal{S}$ should be as dissimilar as possible to the clusters already in $\mathcal{S}$. The key intuition is that candidate clusters in $\mathcal{C}$ that are dissimilar to clusters in $\mathcal{S}$ are those more likely to contain authorship records associated with authors not in $\mathcal{S}$.

In order to compute the similarity among clusters, we employed the well-known cosine function [2]. Specifically, each authorship record is represented as a feature vector, where each coauthor name and each word of work title and publication venue title is a feature, and the similarity is calculated using the centroid of the vectors (i.e., authorship records) in the clusters. The identification of fragmented clusters continues by evaluating the next candidate cluster $c \in \mathcal{C}$ to be inserted into $\mathcal{S}$. Candidate $c \in \mathcal{C}$ is inserted into $\mathcal{S}$ if:

$$\forall s \in \mathcal{S}, \phi(c, s) \leq \phi_{max}$$

That is, c is inserted into $\mathcal{S}$ if the similarity value $\phi(c, s)$ between c and each cluster $s \in \mathcal{S}$ is at most ϕ_{max} (which is a user-specified threshold). Then, c is removed from $\mathcal{C}$ and the iteration continues with the next candidate cluster in $\mathcal{C}$. The process finally stops when there is no more candidate clusters, and $\mathcal{C}$ is finally empty. In the end, authorship records in each cluster $s \in \mathcal{S}$ are inserted into the training data $\mathcal{D}$. Each authorship record receives the author label of the corresponding cluster. The remaining authorship records that were not included in $\mathcal{D}$ (i.e., records associated with fragmented clusters that were not included in $\mathcal{S}$) compose the test set $\mathcal{T}$.

3.2 The Supervised Step

The unsupervised step produces a set of examples, $\mathcal{D}$, which is used to produce a disambiguation function that relates record features to the correct author. Our disambiguation function is a function from $\{f_1, f_2, \ldots, f_m\}$ to $\{a_1, a_2, \ldots, a_n\}$ that is used to predict the correct author for authorship records in the test set $\mathcal{T}$. In the following we describe a supervised method to produce disambiguation functions from $\mathcal{D}$.

Associative Name Disambiguation

The proposed method for supervised author name disambiguation exploits the fact that, frequently, there are strong associations between features $\{f_1, f_2, \ldots, f_m\}$ and specific authors $\{a_1, a_2, \ldots, a_n\}$. The proposed method uncovers such associations from $\mathcal{D}$, and then produces a disambiguation function $\{f_1, f_2, \ldots, f_m\} \rightarrow \{a_1, a_2, \ldots, a_n\}$ using such associations [39]. Typically, these associations are expressed using rules[7] of the form $\mathcal{X} \rightarrow a_1$, $\mathcal{X} \rightarrow a_2$, ..., $\mathcal{X} \rightarrow a_n$, where $\mathcal{X} \subseteq \{f_1, f_2, \ldots, f_m\}$. In the following discussion we denote as $\mathcal{R}$ an arbitrary rule set. Similarly, we denote as $\mathcal{R}_{a_i}$ a subset of $\mathcal{R}$ that is composed of rules of the form $\mathcal{X} \rightarrow a_i$ (i.e., rules predicting author a_i). A rule $\mathcal{X} \rightarrow a_i$ is said to match an authorship record x if $\mathcal{X} \subseteq x$ (i.e., x contains all features in $\mathcal{X}$) and this rule is included in $\mathcal{R}^x_{a_i}$. That is, $\mathcal{R}^x_{a_i}$ is composed of rules predicting author a_i and matching authorship record x. Obviously, $\mathcal{R}^x_{a_i} \subseteq \mathcal{R}_{a_i} \subseteq \mathcal{R}$.

Demand-Driven Rule Extraction

Rule extraction is a major issue for associative name disambiguation, since the number of extracted rules may increase exponentially with the number of features in the training data. The proposed method, on the other hand, extracts rules from the training data on a demand-driven fashion [38], at disambiguation time. The method projects the search space for rules according to information in authorship records in $\mathcal{T}$, allowing rule extraction with efficiency. In other words, the proposed method projects/filters the training data according to the features in authorship record $x \in \mathcal{T}$, and extracts rules from this projected training data, which is denoted as $\mathcal{D}^x$. This ensures that only rules that carry information about record x are extracted from the training data, drastically bounding the number of possible rules.

Prediction

Naturally, there is a total ordering among rules, in the sense that some rules show stronger associations than others. A widely used statistic, called confidence [1] (denoted as $\theta(\mathcal{X} \rightarrow a_i)$), measures the strength of the association between $\mathcal{X}$ and a_i. Put simple, the confidence of the rule $\mathcal{X} \rightarrow a_i$ is given by the conditional probability of a_i being the author of record x, given that $\mathcal{X} \subseteq x$.

Using a single rule to predict the correct author may be prone to error. Instead, the probability (or likelihood) of a_i being the author of record x is estimated by combining rules in $\mathcal{R}^x_{a_i}$. More specifically, $\mathcal{R}^x_{a_i}$ is interpreted as a poll, in which each rule $\mathcal{X} \rightarrow a_i \in \mathcal{R}^x_{a_i}$ is a vote given by features in $\mathcal{X}$ for author a_i. The weight of a vote $\mathcal{X} \rightarrow a_i$ depends on the strength of the association between $\mathcal{X}$ and a_i, which is given by $\theta(\mathcal{X} \rightarrow a_i)$. The process of estimating the probability of a_i being the author of record x starts by summing weighted votes for a_i and then averaging the obtained value by the total number of votes for a_i, as expressed by the score function $s(a_i, x)$ shown in Equation 1 (where $r_j \subseteq \mathcal{R}^x_{a_i}$ and $|\mathcal{R}^x_{a_i}|$ is the number of rules in $\mathcal{R}^x_{a_i}$). Thus, $s(a_i, x)$ gives the average confidence of the rules in $\mathcal{R}^x_{a_i}$ (obviously, the higher the confidence, the stronger the evidence of authorship).

$$s(a_i, x) = \frac{\sum_{j=1}^{|\mathcal{R}^x_{a_i}|} \theta(r_j)}{|\mathcal{R}^x_{a_i}|} \tag{1}$$

The estimated probability of a_i being the author of record x, denoted as $\hat{p}(a_i|x)$, is simply obtained by normalizing $s(a_i, x)$, as shown in Equation 2. A higher value of $\hat{p}(a_i|x)$ indicates a higher likelihood of a_i being the author of x. The author associated with the highest likelihood is finally predicted as the author of record x.

[7] These rules can be efficiently extracted from $\mathcal{D}$ using the strategy proposed in [39].

$$\hat{p}(a_i|x) = \frac{s(a_i, x)}{\sum_{j=1}^{n} s(a_j, x)} \qquad (2)$$

Exploiting Reliable Predictions

Additional examples may be obtained from the predictions performed using the disambiguation function. In this case, reliable predictions are regarded as correct ones and, thus, they can be safely included in the training examples. Next we define the *reliability* of a prediction.

Given an arbitrary authorship record $x \in \mathcal{T}$, and the two most likely authors for x, a_i and a_j, we denote as $\Delta(x)$ the reliability of predicting a_i, as shown in Equation 3.

$$\Delta(x) = \frac{\hat{p}(a_i|x)}{\hat{p}(a_j|x)} \qquad (3)$$

The idea is to only predict a_i if $\Delta(x) \geq \Delta_{min}$, where Δ_{min} is a threshold that indicates the minimum reliability necessary to regard the corresponding prediction as correct, and, therefore, to include it into the training data $\mathcal{D}$. An appropriate value for Δ_{min} can be obtained by performing cross-validation [13], which is a way to predict the fit of a disambiguation function to a hypothetical validation set.

Temporary Abstention

Naturally, some predictions are not enough reliable for certain values of Δ_{min}. An alternative is to abstain from such doubtful predictions. As new examples are included into $\mathcal{D}$ (i.e., the reliable predictions), new evidence may be exploited, hopefully increasing the reliability of the predictions that were previously abstained. To optimize the usage of reliable predictions, we place authorship records in a queue, so that authorship records associated with reliable predictions are considered first. The process works as follows. Initially, authorship records in the test set are randomly placed in the queue. If the author of the record that is located in the beginning of the queue can be reliably predicted, then the prediction is performed, the record is removed from the queue and included into $\mathcal{D}$ as a new example. Otherwise, if the prediction is not reliable, the corresponding authorship record is simply placed in the end of the queue and will be only processed after all other authorship records. The process continues performing more reliable predictions first, until no more reliable predictions are possible. The remaining records in $\mathcal{T}$ (for which only doubtful predictions are possible) are then processed normally, but the corresponding predictions are not included into $\mathcal{D}$. The process stops after all authorship records in $\mathcal{T}$ are processed.

Detecting Novel Authors

We propose to use the lack of rules supporting any already seen author (i.e., authors that are present in some record in $\mathcal{D}$) as evidence indicating the appearance of an unseen author. The number of rules that is necessary to consider an author as an already seen one is controlled by a parameter, γ_{min}. Specifically, for an authorship record $x \in \mathcal{T}$, if the number of rules extracted from $\mathcal{D}^x$ (which is denoted as $\gamma(x)$) is smaller than γ_{min} (i.e., $\gamma(x) < \gamma_{min}$), then the author of x is considered as a new/unseen author and a new label a_k is created to identify such author. Further, this prediction is considered as a new example and included into $\mathcal{D}$. An appropriate value for γ_{min} can be obtained by performing cross-validation.

4. EVALUATION

In this section we present experimental results that demonstrate the effectiveness of our proposed disambiguation method, SAND. We first describe the collection employed, evaluation metrics and baselines. Then, we discuss the performance of the proposed method in these collections.

4.1 Collections

We used collections of authorship records extracted from two digital libraries: DBLP and BDBComp. These collections contain several ambiguous groups.

The collection of authorship records extracted from DBLP sums up 4,287 records associated with 220 distinct authors, which means an average of approximately 20 records per author. This collection includes 2,270 records whose author names are in short format. Small variations of this collection have been used in several other works [14, 16, 15, 28, 41]. Its original version was created by Han et al. [14], and they manually labeled the authorship records. For this, they used the author's publication home page, affiliation name, e-mail, and coauthor names in a complete name format, and also sent emails to some authors to confirm their authorship. The records for which they had insufficient information to be judged were eliminated. Han et al. [14] also replaced the abbreviated publication venue titles by their complete version obtained from DBLP. We used 11 ambiguous groups extracted by Han et al. [14] with some corrections.

Table 1: The DBLP and BDBComp Collections

DBLP		BDBComp	
Ambiguous Group	#Records/ #Authors	Ambiguous Group	#Records/ #Authors
A. Gupta	576/26	A. Oliveira	52/16
A. Kumar	243/14	A. Silva	64/32
C. Chen	798/60	F. Silva	26/20
D. Johnson	368/15	J. Oliveira	48/18
J. Martin	112/16	J. Silva	36/17
J. Robinson	171/12	J. Souza	35/11
J. Smith	921/29	L. Silva	33/18
K. Tanaka	280/10	M. Silva	21/16
M. Brown	153/13	R. Santos	20/16
M. Jones	260/13	R. Silva	28/20
M. Miller	405/12	–	–

The collection of authorship records extracted from BDBComp sums up 363 records associated with 184 distinct authors, approximately 2 records per author. Notice that, although much smaller than the DBLP collection, this collection is very difficult to disambiguate, because it has many authors with only one authorship record. This collection was created by us and contains the 10 largest ambiguous groups found in BDBComp.

Table 1 shows more detailed information about the collections and its ambiguous groups. Disambiguation is particularly difficult in ambiguous groups such as the "C. Chen" group, in which the correct author must be selected from 60 possible authors, and the "F. Silva" group, in which the majority of authors has appeared in only one authorship record.

As mentioned before, each authorship record consists of the author name, the title of the work, the author name and a list of coauthor names, and the title of the publication venue (conference or journal). Other sources of information are also stored (i.e., identifiers for the citation records and the position of each author name within a record), but they are not used for the sake of disambiguation. Pre-processing involved standardizing coauthor names using only the initial letter of the first name along with the full last name, removing punctuation and stop-words of publication and venue titles, and stemming publication and venue titles using Porter's algorithm [29].

4.2 Evaluation Metrics

In order to evaluate the effectiveness of the proposed disambiguation method, we used two evaluation metrics: the K metric, and pairwise F1. These metrics allow us to compare different disambiguation methods under a number of different criteria, which is not usually done in the literature.

In the discussion that follows, we describe these metrics. The key idea is to compare the clusters extracted by disambiguation methods against ideal, perfect clusters, which were manually extracted. Hereafter, a cluster extracted by a disambiguation method will be referred to as *empirical cluster*, while a perfect cluster will be referred to as *theoretical cluster*.

K Metric

The K metric [21] determines the trade-off between the average cluster purity (ACP) and the average author purity (AAP). Given an ambiguous group, ACP evaluates the purity of the empirical clusters with respect to the theoretical clusters for this ambiguous group. Thus, if the empirical clusters are pure (i.e., they contain only authorship records associated with the same author), the corresponding ACP value will be 1. ACP is defined in Equation 4:

$$\mathrm{ACP} = \frac{1}{N}\sum_{i=1}^{e}\sum_{j=1}^{t}\frac{n_{ij}^2}{n_i} \tag{4}$$

where N is the total number of authorship records in the ambiguous group, t is the number of theoretical clusters in the ambiguous group, e is the number of empirical clusters for this ambiguous group, n_i is the total number of authorship records in the empirical cluster i, and n_{ij} is the total number of authorship records in the empirical cluster i which are also in the theoretical cluster j.

For a given ambiguous group, AAP evaluates the fragmentation of the empirical clusters with respect to the theoretical clusters. If the empirical clusters are not fragmented, the corresponding AAP value will be 1. AAP is defined in Equation 5:

$$\mathrm{AAP} = \frac{1}{N}\sum_{j=1}^{t}\sum_{i=1}^{e}\frac{n_{ij}^2}{n_j} \tag{5}$$

where n_j is the total number of authorship records in the theoretical cluster j.

The K metric consists of the geometric mean between ACP and AAP values. It evaluates the purity and fragmentation of the empirical clusters extracted by each method. The K metric is given in Equation 6:

$$K = \sqrt{\mathrm{ACP} \times \mathrm{AAP}} \tag{6}$$

Pairwise F1

Pairwise F1 (pF1) is the F1 metric [30] calculated using pairwise precision and pairwise recall. Pairwise precision (pP) is calculated as $pP{=}\frac{a}{a+c}$, where a is the number of pairs of authorship records in an empirical cluster that are (correctly) associated with the same author, and c is the number of pairs of authorship records in an empirical cluster not corresponding to the same author. Pairwise recall (pR) is calculated as $pR{=}\frac{a}{a+b}$, where b is the number of pairs of authorship records associated with the same author that are not in the same empirical cluster. The F1-metric is defined in Equation 7:

$$p\mathrm{F1} = 2 \times \frac{pP \times pR}{pP + pR} \tag{7}$$

4.3 Baselines

We used four baselines in our experiments: two supervised and two unsupervised methods. The two supervised methods are the ones proposed by Han et al. in [14]. The first method, referred to as NB, uses the naïve Bayes probability model [24], and the second one, referred to as SVM, uses Support Vector Machines (SVMs) [7].

The first considered unsupervised method [16], referred to as KWAY, uses the cosine similarity function between records as a similarity function and follows a clustering approach based on the K-way spectral clustering technique[8], and the second unsupervised method [17], referred to as SVM-DBSCAN, uses a clustering strategy based on the DBSCAN method [11]. We notice that these four methods are representative of the two main types of solution for the disambiguation task and, more importantly, use the same set of bibliographic attributes as our method, allowing a fair and direct comparison.

It is worth mentioning that both steps of SAND (i.e., the unsupervised and the supervised steps described in Section 3) may also be applied in isolation as a final solution by itself. Thus, in addition to compare SAND against the baselines, we are also interested in performing an investigation of the disambiguation performance of each step of SAND when used in isolation. This allows us to have a better understanding of how each step impacts the performance of SAND.

4.4 Results

Experiments were conducted within each ambiguous group. Unless otherwise stated, the values for Δ_{min} and γ_{min}, used in the second step of SAND, were set automatically by performing 5-fold cross-validation using the training data obtained during the first step of SAND. Thus, the only user-defined parameter is ϕ_{max}. The results are compared using statistical significance tests (t-test) with a 99% confidence interval.

[8] The KWAY method does not use training data, however, it assumes that the number of correct clusters is known in advance (i.e., the number of empirical and theoretical clusters is the same).

Table 2: Results (with their standard deviations) obtained by the unsupervised step for each ambiguous group on the DBLP collection.

Ambiguous Group	ACP	AAP	K	pP	pR	pF1
A Gupta	0.990 ± 0.002	0.416 ± 0.033	0.641 ± 0.025	0.994 ± 0.001	0.398 ± 0.056	0.567 ± 0.058
A Kumar	0.995 ± 0.003	0.242 ± 0.011	0.490 ± 0.011	0.995 ± 0.003	0.098 ± 0.006	0.178 ± 0.010
C Chen	0.953 ± 0.003	0.202 ± 0.003	0.439 ± 0.003	0.906 ± 0.008	0.050 ± 0.001	0.095 ± 0.002
D Johnson	1.000 ± 0.000	0.301 ± 0.008	0.548 ± 0.008	1.000 ± 0.000	0.295 ± 0.016	0.455 ± 0.019
J Martin	0.987 ± 0.007	0.500 ± 0.007	0.702 ± 0.007	0.957 ± 0.023	0.322 ± 0.005	0.482 ± 0.008
J Robinson	1.000 ± 0.000	0.355 ± 0.007	0.596 ± 0.005	1.000 ± 0.000	0.285 ± 0.010	0.443 ± 0.011
J Smith	0.971 ± 0.007	0.263 ± 0.031	0.504 ± 0.032	0.982 ± 0.018	0.279 ± 0.054	0.432 ± 0.067
K Tanaka	1.000 ± 0.000	0.380 ± 0.008	0.616 ± 0.006	1.000 ± 0.000	0.231 ± 0.008	0.375 ± 0.011
M Brown	1.000 ± 0.000	0.395 ± 0.007	0.629 ± 0.006	1.000 ± 0.000	0.340 ± 0.013	0.507 ± 0.015
M Jones	1.000 ± 0.000	0.281 ± 0.015	0.530 ± 0.014	1.000 ± 0.000	0.251 ± 0.021	0.400 ± 0.026
M Miller	0.991 ± 0.005	0.603 ± 0.026	0.773 ± 0.017	0.988 ± 0.009	0.586 ± 0.034	0.735 ± 0.026
Average	0.990 ± 0.002	0.358 ± 0.014	0.588 ± 0.012	0.984 ± 0.006	0.285 ± 0.020	0.425 ± 0.023

Table 3: Results (with their standard deviations) obtained by the supervised step for each ambiguous group on the DBLP collection when no training data is provided.

Ambiguous Group	ACP	AAP	K	pP	pR	pF1
A Gupta	0.343 ± 0.099	0.813 ± 0.037	0.521 ± 0.071	0.235 ± 0.089	0.818 ± 0.037	0.356 ± 0.108
A Kumar	0.590 ± 0.110	0.825 ± 0.024	0.695 ± 0.066	0.530 ± 0.137	0.860 ± 0.030	0.645 ± 0.113
C Chen	0.140 ± 0.040	0.858 ± 0.078	0.341 ± 0.037	0.070 ± 0.022	0.841 ± 0.114	0.127 ± 0.033
D Johnson	0.451 ± 0.025	0.823 ± 0.038	0.608 ± 0.016	0.374 ± 0.034	0.895 ± 0.033	0.527 ± 0.035
J Martin	0.458 ± 0.079	0.835 ± 0.035	0.616 ± 0.061	0.318 ± 0.118	0.800 ± 0.056	0.444 ± 0.133
J Robinson	0.570 ± 0.064	0.796 ± 0.037	0.672 ± 0.040	0.458 ± 0.118	0.806 ± 0.064	0.576 ± 0.105
J Smith	0.282 ± 0.071	0.813 ± 0.075	0.472 ± 0.051	0.198 ± 0.069	0.837 ± 0.079	0.312 ± 0.081
K Tanaka	0.466 ± 0.038	0.828 ± 0.042	0.620 ± 0.021	0.342 ± 0.025	0.879 ± 0.034	0.491 ± 0.026
M Brown	0.528 ± 0.109	0.750 ± 0.065	0.627 ± 0.078	0.370 ± 0.163	0.737 ± 0.123	0.478 ± 0.164
M Jones	0.478 ± 0.071	0.729 ± 0.059	0.588 ± 0.046	0.381 ± 0.100	0.707 ± 0.084	0.485 ± 0.078
M Miller	0.562 ± 0.145	0.799 ± 0.066	0.663 ± 0.074	0.522 ± 0.158	0.849 ± 0.057	0.632 ± 0.112
Average	0.442 ± 0.077	0.806 ± 0.051	0.584 ± 0.051	0.345 ± 0.094	0.821 ± 0.065	0.461 ± 0.090

Each competing method was executed ten times, and in each execution a different shuffling configuration was used (i.e., the performance of the evaluated methods may be impacted by the order in which authorship records are processed). The final disambiguation performance in each ambiguous group is given by the average performance over the ten executions. Results regarding the comparison between methods are presented using the average of the final results for each ambiguous group.

How effective are the two steps in isolation (i.e., the unsupervised and supervised steps of SAND), when no training data is available?

We show the results obtained by each step of SAND when applied in isolation on DBLP collection. Table 2 shows the results obtained with the application of the proposed unsupervised method, while Table 3 shows the results obtained with the application of the proposed supervised method. Since there is no training data available for the supervised step, we were unable to perform cross-validation in order to find the parameters. Thus, we set Δ_{min} to 1.5 and γ_{min} to 20, which are their default values. As authorship records in the test set are processed, the supervised method exploits novel authors and reliable predictions in order to produce its own training examples.

We notice that the application in isolation of the unsupervised method leads to extremely pure, but fragmented clusters. We also notice that the application in isolation of the supervised method without any training examples leads to clusters with reduced fragmentation, but also with decreased purity. This complementary behaviour was one of the motivations to combine these two methods.

How the performance of SAND is affected by ϕ_{max}?

Lower values of ϕ_{max} result in the selection of only few clusters, that is, important clusters may be not included in the training data. On the other hand, higher values of ϕ_{max} may result in the selection of several fragmented clusters. Thus, this compromise must be taken into account in order to choose a suitable value for ϕ_{max}. After inspecting typical similarity values using some authorship records, our intuition suggested that a suitable value for ϕ_{max} should be greater than 0.10, and thus we set ϕ_{max} to 0.12.

Figure 2 shows the disambiguation performance of SAND for various values of ϕ_{max}. Clearly, there are several values for ϕ_{max} that lead to results that are better than the results obtained using ϕ_{max}=0.12. However, Figure 2 also shows that setting ϕ_{max}=0.12 is sufficient to ensure a reasonable amount of training examples. All the results to be

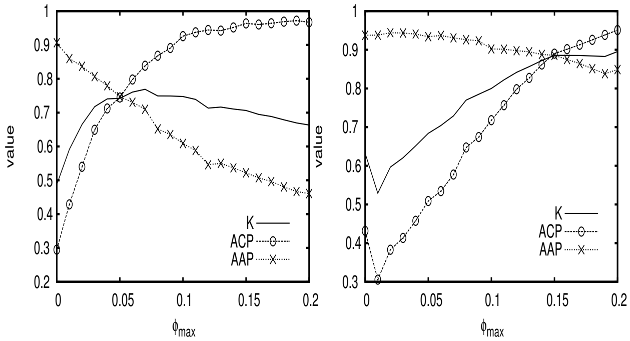

Figure 2: Sensitivity analysis for ϕ_{max}.

shown hereafter concerning the performance of SAND were obtained using ϕ_{max}=0.12.

How effective is SAND?

Table 4 shows the disambiguation performance for SAND in each ambiguous group (i.e., averaged over the 10 executions). As we can see, the standard deviations of these results are very low, meaning that the shuffling configuration, that is, the order in which authorship records are processed, does not affect much the results. Further, Table 4 also shows that groups such as "C. Chen" and "R. Silva" are harder to disambiguate, and this is because these groups contain a large number of candidate authors.

Table 4: Results obtained by the SAND.

Ambiguous Group	K	pF1
A. Gupta	0.753±0.056	0.725±0.077
A. Kumar	0.611±0.015	0.378±0.039
C. Chen	0.538±0.011	0.266±0.016
D. Johnson	0.639±0.050	0.620±0.074
J. Martin	0.790±0.013	0.697±0.030
J. Robinson	0.683±0.043	0.594±0.070
J. Smith	0.733±0.009	0.758±0.011
K. Tanaka	0.699±0.037	0.570±0.057
M. Brown	0.786±0.025	0.759±0.034
M. Jones	0.715±0.060	0.679±0.082
M. Miller	0.881±0.005	0.887±0.005
Average	0.712±0.009	0.630±0.015

(a) DBLP

Ambiguous Group	K	pF1
A Oliveira	0.842±0.010	0.811±0.021
A Silva	0.814±0.012	0.678±0.028
F Silva	0.885±0.009	0.544±0.025
J Oliveira	0.817±0.025	0.711±0.048
J Silva	0.856±0.025	0.674±0.067
J Souza	0.850±0.012	0.718±0.017
L Silva	0.851±0.012	0.658±0.032
M Silva	0.811±0.002	0.366±0.012
R Santos	0.942±0.000	0.571±0.000
R Silva	0.750±0.004	0.235±0.012
Average	0.842±0.004	0.597±0.008

(b) BDBComp

How effective is SAND when compared against unsupervised methods?

We show results concerning the comparison of the disambiguation performance obtained by SAND and the performance obtained by the two unsupervised baseline methods, KWAY and SVM-DBSCAN. For KWAY, we used the implementation of the K-way spectral clustering provided by the University of Washington spectral clustering working group[9]. For SVM-DBSCAN, we used LibSVM package [6] and the DBSCAN version available from Weka[10].

Table 5: Results obtained by the SAND, KWAY and SVM-DBSCAN methods. Best results are highlighted in bold.

Method	K	pF1
SAND	**0.712**	**0.630**
KWAY	0.560	0.402
SVM-DBSCAN	0.406	0.279

(a) DBLP

Method	K	pF1
SAND	**0.842**	**0.597**
KWAY	0.805	0.436
SVM-DBSCAN	0.339	0.088

(b) BDBComp

As we can see in Table 5, in the DBLP collection, SAND outperforms KWAY and SVM-DBSCAN methods, providing gains of more than 27% under the K metric and 56% under the pF1 metric. In the BDBComp collection, SAND outperforms the KWAY and SVM-DBSCAN methods by more than 36% under the pF1 metric.

The poor performance of SVM-DBSCAN is mainly due to the small number of attributes used when compared with the original proposed method described in [17]. In that work, several other attributes such as affiliation and e-mail were used. In this scenario of author name disambiguation where only few attributes are available, the similarity functions learned by the SVM-DBSCAN are not able to generalize suitably. The KWAY method, on the other hand, exploits only the similarity between records to group them. Thus, it might be able to create better clusters than SVM-DBSCAN, however, possibly incurring in more errors (i.e., wrong assignments). SAND produced better results since it is able to predict the author of a given record using disambiguation functions learned from examples automatically selected.

How effective is SAND when compared against supervised methods?

To check whether our associative name disambiguator is the best choice for the supervised step, we compared the disambiguation performance of SAND with alternative self-training methods (i.e., we used other learning algorithms in the supervised step). We evaluated SVM and Naïve Bayes techniques on the DBLP and BDBComp collections. Table 6 shows the results. We notice that SAND outperforms all competitors. The superiority of SAND is mostly due to its capability of adding new examples based on reliable pre-

[9] `http://www.stat.washington.edu/spectral`

[10] `http://www.cs.waikato.ac.nz/ml/weka/`

dictions, and also of identifying new authors not present in the provided training data.

Table 6: Results obtained by the self-training approach (i.e., the unsupervised step) coupled with SVMs (S-SVM) and Naïve Bayes (S-NB) techniques in the second step (i.e., the supervised step). Best results are highlighted in bold.

Method	K	pF1
SAND	**0.712**	**0.630**
S-SVM	0.592	0.444
S-NB	0.591	0.449

(a) DBLP

Method	K	pF1
SAND	**0.842**	**0.597**
S-SVM	0.825	0.479
S-NB	0.820	0.457

(b) BDBComp

In order to assess the performance of our disambiguator even further, we conducted another experiment in which we selected part of the collection as training data. For this experiment, we used 5-fold cross-validation for the supervised methods. It is very important to notice that the performance of our method in this experiment corresponds to the performance only in test set (one fold) and that we did not use the training examples. As expected, on the DBLP collection the supervised methods achieve better results since about 80% of the collection was used for training. However, the loss in performance of SAND when compared to the best supervised method was at most 20% (against SVM), showing that all the effort of labeling a large amount of training data may not be worth it. More important, when we provided the same training data to the second step of SAND, it outperforms all other supervised methods by 6% against SVM and 13% against NB, showing that it is able to better explore the manually provided training data along with its other self-training, transductive characteristics. On the BDBComp collection, SAND outperforms all methods under all metrics by more than 60%. Due to space limitations, the discussion of a more elaborated version of this experiment, including, for instance, the investigation of other training data rates, is out of the scope of this paper and left for future work.

5. CONCLUSIONS AND FUTURE WORK

Name disambiguation, in the context of bibliographic citations, is the task of determining whether records in a collection of publications refer to the same person. This problem is widespread in many large-scale digital libraries, such as Citeseer, Google Scholar and DBLP, and it is particularly acute when citations are built automatically.

In this work, we proposed a method, called SAND (standing for Self-training Associative Name Disambiguator), that follows a two-step approach for name disambiguation. In the first step (i.e., the unsupervised step) a set of authorship records are clustered so that records that are likely to be associated with the same author are grouped together in clusters and some of these clusters are selected to be used as training data. In the second step (i.e., the supervised step), these selected clusters are used as training data and are given as input to a associative name disambiguator with the ability to detect the appearance of new authors that were not included in the training data.

We used two collections, one extracted from the DBLP and the other from BDBComp digital libraries to evaluate SAND. On the DBLP collection, SAND outperformed two unsupervised methods in more than 27%. On the BDBComp collection, SAND outperformed two unsupervised methods in more than 36% under the pF1 metric and in more 4% under the K metric. Our experimental results also show that: (1) there is some sensitivity of the method to the choice of the user-defined parameter, ϕ_{max}, although there are some ranges of values in which the results are very stable and (2) the combination of the first step of our method with other supervised ones does not produce good results as we obtained with SAND.

As future work, we intend to investigate other approaches to select the clusters to be used as training data, to perform experiments on other collections and to develop strategies to free our method from any parameter setup.

Acknowledgments

This research is partially funded by the MCT/CNPq/CT-INFO project InfoWeb (grant number 55.0874/2007-0), by InWeb - The National Institute of Science and Technology for the Web (MCT/CNPq/FAPEMIG grant number 573871/2008-6), and by the authors's individual research grants from CAPES, CNPq, and FAPEMIG (proj. 14281).

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of SIGMOD*, pages 207–216. ACM, 1993.

[2] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[3] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *Proc. of WWW*, pages 463–470, Chiba, Japan, 2005. ACM.

[4] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, Bethesda, MD, USA, 2006.

[5] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1):5, 2007.

[6] C.-C. Chang and C.-J. Lin. *LibSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[8] R. G. Cota, M. A. Gonçalves, and A. H. F. Laender. A heuristic-based hierarchical clustering method for author name disambiguation in digital libraries. In *Proc. of SBBD*, pages 20–34, João Pessoa, Paraiba, Brazil, 2007.

[9] A. Culotta, P. Kanani, R. Hall, M. Wick, and A. McCallum. Author disambiguation using

error-driven machine learning with a ranking loss function. In *Sixth International Workshop on Information Integration on the Web*, Vancouver, Canada, 2007.

[10] C. P. Diehl, L. Getoor, and G. Namata. Name reference resolution in organizational email archives. In *Proc. of the SIAM Intl. Conf. on Data Mining*, pages 70–91, Bethesda, MD, USA, 2006.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD*, pages 226–231, Portland, Oregon, 1996. AAAI Press.

[12] C. Galvez and F. de Moya Anegón. Approximate personal name-matching through finite-state graphs. *Journal of the American Society for Information Science and Technology*, 58(13):1960–1976, 2007.

[13] S. Geisser. *Predictive inference: An introduction.* Chapman & Hall, New York, 1993.

[14] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Proc. of JCDL*, pages 296–305, Tucson, AZ, USA, 2004. ACM.

[15] H. Han, W. Xu, H. Zha, and C. L. Giles. A hierarchical naive Bayes mixture model for name disambiguation in author citations. In *Proc. of SAC*, pages 1065–1069, Santa Fe, New Mexico, 2005. ACM.

[16] H. Han, H. Zha, and C. L. Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *Proc. of JCDL*, pages 334–343, Denver, CO, USA, 2005. ACM.

[17] J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. In *Proc. of PKDD*, pages 536–544, Berlin, Germany, 2006. Springer.

[18] P. Kanani, A. McCallum, and C. Pal. Improving author coreference by resource-bounded information gathering from the web. In *Proc. of IJCAI*, pages 429–434, Hyderabad, India, 2007.

[19] I.-S. Kang, S.-H. Na, S. Lee, H. Jung, P. Kim, W.-K. Sung, and J.-H. Lee. On co-authorship for author disambiguation. *Information Processing & Management*, 45(1):84–97, 2009.

[20] A. H. F. Laender, M. A. Gonçalves, R. G. Cota, A. A. Ferreira, R. L. T. Santos, and A. J. C. Silva. Keeping a digital library clean: new solutions to old problems. In *Proc. of DocEng*, pages 257–262, 2008.

[21] I. Lapidot. Self-Organizing-Maps with BIC for Speaker Clustering. Technical report, IDIAP Research Institute, Martigny, Switzerland, 2002.

[22] D. Lee, J. Kang, P. Mitra, C. L. Giles, and B.-W. On. Are your citations clean? *Communications of the ACM*, 50(12):33–38, 2007.

[23] B. Malin. Unsupervised name disambiguation via social network similarity. In *Proc. of the Workshop on Link Analysis, Counterterrorism, and Security*, pages 93–102, Newport Beach, CA, 2005.

[24] T. M. Mitchell. *Machine Learning.* McGraw-Hill, New York, NY, USA, 1997.

[25] B.-W. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. An effective approach to entity resolution problem using quasi-clique and its application to digital libraries. In *Proc. of JCDL*, pages 51–52, Chapel Hill, NC, USA, 2006. ACM.

[26] B.-W. On and D. Lee. Scalable name disambiguation using multi-level graph partition. In *Proc. of the SDM Conf.*, Minneapolis, Minnesota, USA, 2007. SIAM.

[27] B.-W. On, D. Lee, J. Kang, and P. Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *Proc. of JCDL*, pages 344–353, Denver, CO, USA, 2005.

[28] D. A. Pereira, B. A. Ribeiro-Neto, N. Ziviani, A. H. F. Laender, M. A. Gonçalves, and A. A. Ferreira. Using web information for author name disambiguation. In *Proc. of JCDL*, pages 49–58, Austin, TX, USA, 2009.

[29] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[30] C. J. V. Rijsbergen. *Information Retrieval, 2nd edition.* Butterworths, London, 1979.

[31] C. L. Scoville, E. D. Johnson, and A. L. McConnell. When A. Rose is not A. Rose: the vagaries of author searching. *Medical reference services quarterly*, 22(4):1–11, 2003.

[32] N. R. Smalheiser and V. I. Torvik. *Author Name Disambiguation*, volume 43, pages 287–313. 2009.

[33] J. M. Soler. Separating the articles of authors with the same name. *Scientometrics*, 72(2):281–290, 2007.

[34] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Efficient topic-based unsupervised name disambiguation. In *Proc. of JCDL*, pages 342–351, Vancouver, BC, Canada, 2007. ACM.

[35] V. I. Torvik and N. R. Smalheiser. Author name disambiguation in medline. *ACM Transactions on Knowledge Discovery from Data*, 3(3), 2009.

[36] V. I. Torvik, M. Weeber, D. R. Swanson, and N. R. Smalheiser. A probabilistic similarity metric for Medline records: A model for author name disambiguation. *Journal of the American Society for Information Science and Technology*, 56(2):140–158, 2005.

[37] P. Treeratpituk and C. L. Giles. Disambiguating authors in academic publications using random forests. In *Proc. of JCDL*, pages 39–48, Austin, TX, USA, 2009.

[38] A. Veloso, W. Meira, M. Cristo, M. Gonçalves, and M. Zaki. Multi-evidence, multi-criteria, lazy associative document classification. In *Proc. of CIKM*, pages 218–227. ACM, 2006.

[39] A. Veloso, W. Meira Jr., and M. J. Zaki. Lazy associative classification. In *Proc. of ICDM*, pages 645–654. IEEE, 2006.

[40] Q. M. Vu, T. Masada, A. Takasu, and J. Adachi. Using a knowledge base to disambiguate personal name in web search results. In *Proc. of SAC*, pages 839–843, Seoul, Korea, 2007. ACM.

[41] K.-H. Yang, H.-T. Peng, J.-Y. Jiang, H.-M. Lee, and J.-M. Ho. Author name disambiguation for citations using topic and web correlation. In *Proc. of ECDL*, pages 185–196, Aarhus, Denmark, 2008. Springer-Verlag.