CONTRIBUTIONS FOR SOLVING THE AUTHOR NAME AMBIGUITY PROBLEM IN BIBLIOGRAPHIC CITATIONS

ANDERSON ALMEIDA FERREIRA

CONTRIBUTIONS FOR SOLVING THE AUTHOR NAME AMBIGUITY PROBLEM IN BIBLIOGRAPHIC CITATIONS

Tese apresentada ao Programa de Pós-Graduação em Computer Science do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais – Departamento de Ciência da Computação como requisito parcial para a obtenção do grau de Doutor em Computer Science.

Orientador: Marcos André Gonçalves
Co-orientador: Alberto Henrique Frade Laender

Belo Horizonte

Junho de 2012

ANDERSON ALMEIDA FERREIRA

CONTRIBUTIONS FOR SOLVING THE AUTHOR NAME AMBIGUITY PROBLEM IN BIBLIOGRAPHIC CITATIONS

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais – Departamento de Ciência da Computação in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Marcos André Gonçalves
Co-advisor: Alberto Henrique Frade Laender

Belo Horizonte

June 2012

© 2012, Anderson Almeida Ferreira.
Todos os direitos reservados.

Ferreira, Anderson Almeida
F383c Contributions for Solving the Author Name Ambiguity Problem in Bibliographic Citations / Anderson Almeida Ferreira. — Belo Horizonte, 2012
xx, 114 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais – Departamento de Ciência da Computação
Orientador: Marcos André Gonçalves
Co-orientador: Alberto Henrique Frade Laender

1. Computação – Teses. 2. Sistemas de recuperação da informação – Teses. Bibliotecas digitais – Teses. 3. Registros de autoridade de nomes (Recuperação da informação). I. Orientador. II. Coorientador. III. Titulo.

CDU 519.6*73(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Contribuições para solucionar o problema de ambiguidade de nomes de autores em citações bibliográficas

ANDERSON ALMEIDA FERREIRA

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. MARCOS ANDRÉ GONÇALVES - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ALBERTO HENRIQUE FRADE LAENDER - Co-orientador
Departamento de Ciência da Computação - UFMG

PROF. CARLOS ALBERTO HEUSER
Departamento de Informática - UFRGS

PROF. CLODOVEU AUGUSTO DAVIS JÚNIOR
Departamento de Ciência da Computação - UFMG

PROFA. GISELE LOBO PAPPA
Departamento de Ciência da Computação - UFMG

PROF. RICARDO DA SILVA TORRES
Instituto de Computação - UNICAMP

Belo Horizonte, 21 de junho de 2012.

To Lília, Lucas, André, Geralda and Guimarães.

Acknowledgments

I thank my advisor, professor Marcos André Gonçalves, and my coadvisor, professor Alberto H. F. Laender, for supporting to developed this thesis.

I thank my coauthors that contributed significantly to the development of the articles used as base to this thesis, specially, Marcos, Alberto, Adriano, Jussara, Ana Paula and Rodrigo.

I thank my friends at LBD, the UFMG database group, for the nice environment to work in this laboratory.

I also thank the administrative staffs of PPGCC that always solve all questions with respect my PhD, such as travels, documentation and so on.

Finally, I am grateful to my parents, Guimarães and Geralda, who always encouraged me, to my wife, Lília, for your understanding and sacrifices during my studies, and to my sons, Lucas and André.

This research is partially funded by the InWeb - The National Institute of Science and Technology for the Web (MCT/CNPq/FAPEMIG grant number 573871/2008-6), InfoWeb (MCT/CNPq grant 55.0874/2007-0) and by CNPq and FAPEMIG scholarships. This financial support is gratefully acknowledged.

Abstract

Author name ambiguity is a problem that occurs when a set of bibliographic citation records contains ambiguous author names, i.e., the same author may appear under distinct names, or distinct authors may have similar names. This is one of the hardest problems faced by current scholarly digital libraries (DLs), such as DBLP, CiteSeer, MEDLINE and BDBComp. In this thesis, we present a set of contributions to help solving the author name ambiguity problem. First of all, we present a taxonomy to classify the author name disambiguation methods that helps to better understand how the methods work and consequently understand their limitations. Second, we present SAND a new hybrid disambiguation method that exploits the strengths of both supervised author assignment and unsupervised author grouping methods. SAND is a three-step disambiguation method. In its first step (i.e., the author grouping step), a set of citation records is clustered so that records that are likely to be associated with the same author are grouped together in clusters. In its second step (i.e., the cluster selection step), some of these clusters are selected to be used as training data. Finally, in its third step (i.e., the author assignment 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. As our final contribution, we present SyGAR, a new generator of synthetic citation records that helps to evaluate author name disambiguation methods under several scenarios. SyGAR generates synthetic citation records following the publication profiles of existing authors, extracted from an input collection. Moreover, SyGAR allows the simulation of several real-world scenarios such as the introduction of new authors (not present in the input collection), dynamic changes in an author's publication profile as well as the introduction of typographical errors in the synthetic citations.

List of Figures

List of Tables

Contents

Chapter 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 attributes such as author and coauthor names, work and publication venue titles 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 manner. Also, studies based on DL content can lead to interesting results such as coverage of topics, research tendencies, quality and impact of publications of a specific subcommunity or individuals, patterns of collaboration in social networks, etc. These types of analysis and information, which are used, for instance, by funding agencies on decisions for grants and for individual's promotions, presuppose *high quality content* [Laender et al., 2008; Lee et al., 2007].

According to Lee et al. [2007], the challenges to have high quality content comes from data-entry errors, citation formats, lack of (enforcement of) standards, imperfect citation-gathering software, ambiguous author names, abbreviations of publication venue titles and large-scale citation data.

Among these challenges, the problem of *ambiguous author names* has required a lot of attention from the DL research community due to its inherent difficulty. Specifically, ambiguity of author names is a problem that occurs when a set of citation records contains ambiguous author names, i.e., the same author may appear under distinct names (synonyms), or distinct authors may have similar names (homonyms).

[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

This problem may be caused by a number of reasons [McKay et al., 2010], including name changes due to personal circumstances, variation in transliteration of non-roman names, typographical errors, lack of standards and common practices, and decentralized generation of content (i.e., by means of automatic harvesting [Lagoze and de Sompel, 2001]).

An interesting example that illustrates the author name ambiguity problem can be taken from DBLP. Until recently, if one searched for the author name "Mohammed Zaki", the result would include three name variations - "Mohammed Zaki", "Mohammed J. Zaki" and "Mohammed Javeed Zaki" (see Figure 1.1). Although all these three names seemed to refer to the same person, they in fact illustrate a case that involves both synonyms and homonyms. While the first name referred to Mohammed Zaki from Al-Azhar University, Nasr City, Cairo, Egypt, the second and third names referred to Mohammed Zaki from the Rensselaer Polytechnic Institute Department of Computer Science,USA, thus characterizing a synonym situation.

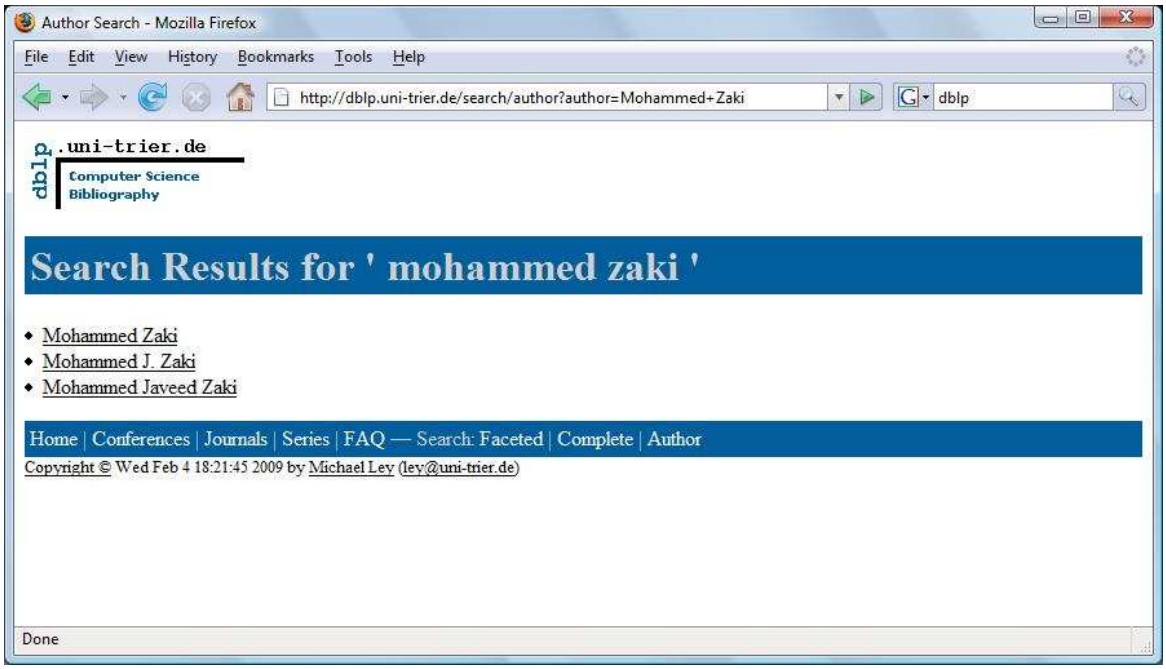


Figure 1.1. Synonyms: a unique author with several name variations.

On the other hand, by clicking on the "Mohammed Zaki" link the resulting page (see Figure 1.2) would show an example of homonym, since the second citation actually corresponds to a paper coauthored by Mohammed Javeed Zaki from the Department of Computer Science, Rensselaer Polytechnic Institute, USA. Although in this case the problem was caused by different variations of an author's names, there are many other

cases in which two different authors simply have the same name, a common situation, for example, for authors with Asian names.

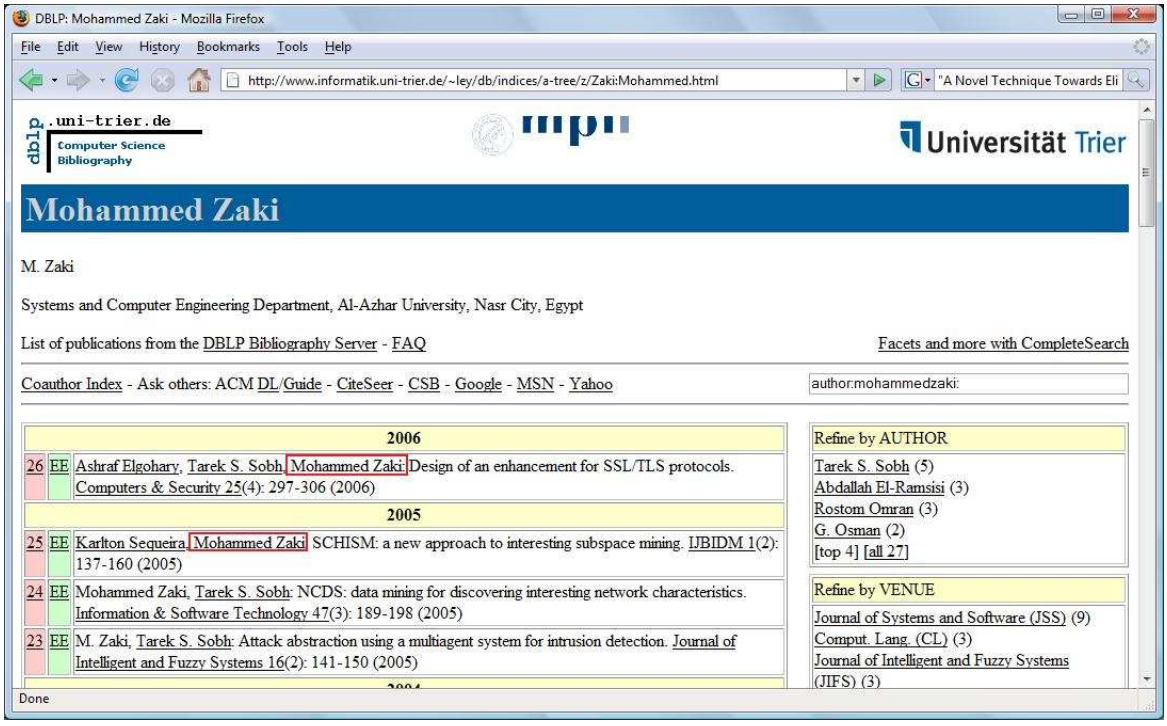


Figure 1.2. Homonyms: several authors with a same name variation.

1.1 Motivation

There are several open challenges that need to be addressed in order to produce more reliable solutions that can be employed in a production mode in real digital libraries. Below we discuss some of them.

Effectiveness. Methods for disambiguating author names must be effective, i.e., they must correctly disambiguate the author names in bibliographic citations. Although many methods have been reported in the literature (see Chapter 3 for a comprehensive coverage of those), there is still a lot of room for improvements.

Very Little Data in the Citations. In most cases we have only basic information about the citations available: author (coauthor) names, work and publication venue titles, and publication year. Furthermore, in some cases author names contain only the initial and the last surname and the publication venue title is abbreviated. New strategies that try to derive implicit information (e.g., topics) or gather additional information from the Web are promising in this scenario.

Very Ambiguous Cases. Several methods exploit coauthor-based heuristics, by explicitly assuming the hypotheses that: (i) very rarely ambiguous references will have coauthors in common who have also ambiguous names; or (ii) it is rare that two authors with very similar names work in the same research area. These hypotheses work in most cases but, when they fail, the errors they generate are very hard to fix. For example, in the case of authors with Asian names, the first hypothesis fails much more frequently than for authors with English or Latin names.

Citations with Errors. Errors occur in citation data which are sometimes impossible to detect. The methods need to be tolerant to such errors.

Efficiency. With the high amount of articles being published nowadays in the different knowledge areas, the solutions need to deal with the problem efficiently. Few proposed methods have this explicit concern.

Practicality and Cost. As we shall see, most of the best current methods for solving the author name disambiguation problem are supervised, i.e., they require large amounts of manually labeled data explicitly indicating whether two ambiguous names correspond to the same author or no, to serve as training for some machine learning procedure [Ferreira et al., 2012b]. Creating such training data is very costly and laborious. This also may hurt the practical application of these methods, mainly as the digital library evolves and more training is required to learn new patterns.

Adaptability to Different Knowledge Areas. As we shall see, most of the collections used to evaluate the methods are related to Computer Science. However, other knowledge areas (e.g., Humanities, Biology, Medicine) may have different publication patterns (e.g., many publications with a sole author or with a lot of coauthors) which may cause some additional difficulties for the current generation of methods, requiring adaptations.

Incremental Disambiguation. Ideally, disambiguation should be performed incrementally as new citations are incorporated into the DL, since it is not reasonable to assume that the whole DL should be disambiguated at each new load. However, most, if not all, methods ignore this fact.

Evaluation. The methods for disambiguating author names in bibliographic citations are usually evaluated in static scenarios without considering a time evolving digital library, containing dynamic patterns such as the introduction of citations of new authors and the change of researchers' interests/expertises over time.

Author Profile Changes. It is very common that the research interests of an author change over time. This can happen for many reasons, for example, new collaborations, change in research group or institution, natural evolution of a research field, etc. These changes may cause modifications in the model representing the author

profile causing difficulties for the methods. A possible solution probably involves retraining, but determining when to retrain is a challenge. However, this issue has been largely ignored by all methods.

New Authors. The methods should be capable of identifying references to new ambiguous authors who do not have citations in the DL yet.

These challenges have led to a myriad of author disambiguation methods [Bhattacharya and Getoor, 2006, 2007; Culotta et al., 2007; Fan et al., 2011; Han et al., 2004, 2005a,b; Huang et al., 2006; Kanani et al., 2007; Kang et al., 2009; Levin and Heuser, 2010; Levin et al., 2012; Malin, 2005; On et al., 2006; Pereira et al., 2009; Shu et al., 2009; Soler, 2007; Song et al., 2007; Tang et al., 2012; Torvik et al., 2005; Treeratpituk and Giles, 2009; Yang et al., 2008]. 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 meaning that there is space for improvements.

In this thesis, we are particularly interested in the *Effectiveness*, *Practicabilty and Cost*, and *Evaluation* challenges. To help with the first two challenges, we propose SAND (standing for Self-training Author Name Disambiguator). As mentioned before, the most effective methods usually follow a *supervised* approach. These methods exploit a set of training examples, from which a disambiguation function is derived and then used to assign the citation records to their corresponding authors. However, the acquisition of training examples requires skilled human annotators to manually label citation records. DLs are very dynamic systems, thus manual labeling of large volumes of examples is unfeasible. On the other hand, *unsupervised* methods require no manual labeling effort, since they simply group citation records into clusters by maximizing intra-cluster similarity while minimizing inter-cluster similarity. SAND exploits the strengths of both supervised and unsupervised methods. Specifically, it works in three steps. In the first step, *(author grouping)*, in an unsupervised way, recurring patterns in the coauthorship graph are exploited in order to produce very pure clusters of references. In the second step, *(cluster selection)*, a subset of the clusters produced in the previous step is selected as training data for the next step. Then, in the third step, *(author assignment)*, a learned function is derived to disambiguate the references in the clusters that were not selected in the previous step.

To help addressing the *Evaluation* challenge, we propose SyGAR (standing for Synthetic Generator of Authorship Records). It is capable of generating synthetic citation records given as input a list of disambiguated records of citations extracted from a real digital library (input collection). The synthetically generated records follow the publication profiles (i.e., distributions of title terms, coauthor names and publication

venue title) of existing authors extracted from the input collection. Moreover, SyGAR can be parameterized to generate records for new authors (not present in the input collection), for authors with dynamic profiles, as well as records containing typographical errors.

1.2 Contributions

The two main hypotheses of this thesis are that we may: (1) automatically select and label the examples used by a supervised technique, aiming to efficiently produce a disambiguation function that will be used to disambiguate the author names in the citation records, and (2) produce realistic collections to evaluate the disambiguation methods in various scenarios. In order to confirm these hypotheses, the main contributions of this thesis are:

1. A taxonomy for classifying author name disambiguation methods [Ferreira et al., 2012b] that allowed us to better understand the current methods proposed in the literature and present a survey of the most representative ones;

2. SAND (standing for Self-training Author Name Disambiguator) [Ferreira et al., 2010], a new hybrid disambiguation method, that exploits the strengths of both unsupervised and supervised techniques for author name disambiguation; and

3. SyGAR (standing for Synthetic Generator of Authorship Records) [Ferreira et al., 2009, 2012a], a new tested and validated synthetic generator of citation records, that helps evaluating, in several realistic scenarios and under controlled conditions, solutions to the name ambiguity problem as well as to other problems related to name ambiguity.

In addition to the above contributions, the work presented in this thesis also influenced the development and evaluation of other methods, namely HHC (Heuristic-based Hierarchical Clustering) [Cota et al., 2010], WAD (Web Author Disambiguation) [Pereira et al., 2009], INDi (Incremental Name Disambiguation) [Carvalho et al., 2011], SSAND (Selective Sampling for Author Name Disambiguation) [Ferreira et al., 2012c] and SLAND (Self-training Lazy Associative Name Disambiguation) [Veloso et al., 2012].

1.3 Thesis Outline

The rest of this thesis is structured in as follows.

Chapter 2 [**The Author Name Disambiguation Task - Foundations**] formally defines the name disambiguation task and some metrics and collections used to evaluate disambiguation methods are presented.

Chapter 3 [**Automatic Author Name Disambiguation Methods**] defines a taxonomy for classifying name disambiguation methods and provide a description of several representative methods.

Chapter 4 [**SAND: Self-training Author Name Disambiguator**] describes our proposed author name disambiguation method along with its evaluation.

Chapter 5 [**SyGAR: Synthetic Generator of Authorship Records**] presents our generator of synthetic citation records to evaluated disambiguation methods.

Chapter 6 [**Conclusion**] concludes the thesis, by summarizing our results and discussing future work.

Chapter 2

The Author Name Disambiguation Task - Foundations

In this chapter, we formally characterize the name disambiguation task and describe some metrics and collections used to evaluate disambiguation methods.

To illustrate the definitions, we will use the examples showed in Table 2.1. In this table there are four citations ($\{c_1, c_2, c_3, c_4\}$), where each one has its author names identified in this table by $r_j, 1 \leq j \leq 20$. The author names r_3 and r_{15} are examples of homonyms where r_3 refers to "Ajay Gupta" from IBM Research, India and r_{15} refers to "Aarti Gupta" from NEC Laboratories America, USA. The names r_3 and r_7 are examples of synonyms. Both names refer to Ajay Gupta from IBM Research - India.

Table 2.1. Illustrative example (ambiguous group of A. Gupta).

Citation Id	Citation
c_1	(r_1) S. Godbole, (r_2) I. Bhattacharya, (r_3) A. Gupta, (r_4) A. Verma. Building re-usable dictionary repositories for real-world text mining. CIKM, 2010.
c_2	(r_5) Indrajit Bhattacharya, (r_6) Shantanu Godbole, (r_7) Ajay Gupta, (r_8) Ashish Verma, (r_9) Jeff Achtermann, (r_{10}) Kevin English. Enabling analysts in managed services for CRM analytics. KDD, 2009.
c_3	(r_{11}) T. Nghiem, (r_{12}) S. Sankaranarayanan, (r_{13}) G. E. Fainekos, (r_{14}) F. Ivancic, (r_{15}) A. Gupta, (r_{16}) G. J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. HSCC, 2010.
c_4	(r_{17}) William R. Harris, (r_{18}) Sriram Sankaranarayanan, (r_{19}) Franjo Ivancic, (r_{20}) Aarti Gupta. Program analysis via satisfiability modulo path programs. POPL, 2010.

2.1 Definitions

We start with some basic definitions.

Definition 2.1.1 (Citation Record) *A citation record c is a set of bibliographic data, such as author names, work title, publication venue title, publication year, etc., that is pertinent to a particular article. More formally, each citation record c has a list of attributes that includes at least author names, work title and publication venue title. A specific value is associated to each attribute in a citation, which may be composed of several elements. In case of the attribute "author names", an element corresponds to the name of a single unique author. In case of the other attributes, an element corresponds to a word/term.*

Definition 2.1.2 (Reference) *Each author name element is a reference r to an author. We associate a list of attributes to each reference r. In the context of bibliographic citations, r.author corresponds to the author name attribute, r.coauthors corresponds to the other author names in a citation record (coauthors), r.title corresponds to the work title attribute, r.venue corresponds to the publication venue title attribute, and other attributes such as publication year, affiliation, e-mail and so on.*

For instance, the reference r_3 in the citation c_1 in the Table 2.1 has the following attributes values: $r_3.author$="A. Gupta", $r_3.coauthors$={"S. Godbole", "I. Bhattacharya", "A. Verma"}, $r_3.title$="Building re-usable dictionary repositories for real-world text mining", $r_3.venue$="CIKM" and $r_3.year$="2010".

Definition 2.1.3 (Ambiguous Group) *An Ambiguous group is a group of references whose value of the author name attribute are ambiguous, i.e., groups of references having author name attributes with similar names.*

2.2 Task Characterization

The name disambiguation task may be formulated as follows: Let $C = \{c_1, c_2, ..., c_k\}$ be a set of citation records. Each element of the attribute "author names" is a reference r_j to an author. The objective of a disambiguation method is to produce a disambiguation function that is used to partition the set of references to authors $\{r_1, r_2, \ldots, r_m\}$ into n sets $\{a_1, a_2, \ldots, a_n\}$, so that each partition a_i contains (all and ideally only all) the references to a same author.

To disambiguate the bibliographic citations of a DL, we should first split the set of references to authors into ambiguous groups. The ambiguous groups may be obtained,

for instance, by using a blocking method [On et al., 2005]. Blocking methods address scalability issues avoiding the need for comparisons among all references.

2.3 Evaluation Metrics

In this section, we describe K, pairwise F1, cluster F1, RCS and B-cubed metrics that are usually used for evaluating disambiguation methods. 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 [Lapidot, 2002] determines the trade-off between the average cluster purity (ACP) and the average author purity (AAP) or *cohesion*. 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 references to the same author), the corresponding ACP value will be 1. ACP is defined in Equation 2.1:

$$\text{ACP} = \frac{1}{N} \sum_{i=1}^{e} \sum_{j=1}^{t} \frac{n_{ij}^2}{n_i} \tag{2.1}$$

where N is the total number of references 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 references in the empirical cluster i, and n_{ij} is the total number of references in the empirical cluster i which are also in the theoretical cluster j.

For a given ambiguous group, the cohesion metric 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. In other words, the cohesion metric AAP can be thought as the inverse of the fragmentation. The higher the AAP value, the less fragmented are the clusters. AAP is defined in Equation 2.2:

$$\text{AAP} = \frac{1}{N} \sum_{j=1}^{t} \sum_{i=1}^{e} \frac{n_{ij}^2}{n_j} \tag{2.2}$$

where n_j is the total number of references 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 2.3:

$$K = \sqrt{\text{ACP} \times \text{AAP}} \tag{2.3}$$

Pairwise F1

Pairwise F1 (pF1) is the F1 metric [Rijsbergen, 1979] 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 references in an empirical cluster that are (correctly) associated with the same author, and c is the number of pairs of references 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 references associated with the same author that are not in the same empirical cluster. The F1-metric is defined in Equation 2.4:

$$p\text{F1} = 2 \times \frac{pP \times pR}{pP + pR} \tag{2.4}$$

Cluster F1

Cluster F1 ($cF1$) is the F1 metric calculated using cluster precision and cluster recall that measures the performance at the cluster level. Cluster precision (cP) is calculated as $cP = a/(a + c)$, where a is the number of completely correct clusters (a correct cluster should have all the references of an author and only them, i.e., none of another author; otherwise it is incorrect) and c is the number of incorrect clusters. Cluster recall (cR) is calculated as $cR = a/(a + b)$, where b is the number of clusters that should be created but were not. This is a metric to summarize information about the completely correct clusters generated by the method. Likewise, cF1 is analogously defined by the above formula.

Ratio of Cluster Size

The ratio of cluster size (RCS) is the number of empirical clusters versus the number of theoretical clusters. This serves to evaluate how close is the measure to the ideal number of clusters to be generated.

B-Cubed

B-Cubed metric was proposed by Bagga and Baldwin [1998] and has been used to evaluate Web person name search task [Artiles et al., 2010]. B-Cubed calculates the final precision and recall based on the precision (P_r) and recall (R_r) of each reference r that are defined as:

$$P_r = \frac{n_i^r}{n_i} \tag{2.5}$$

$$R_r = \frac{n_i^r}{n_j} \tag{2.6}$$

where n_i^r is the total number of references that refer to same author of r and belong to the same empirical cluster i that contains r, n_i is the total number of references in the empirical cluster i that contains r and n_j is the total number of references in the theoretical cluster j that contains r.

The final precision (bP) and recall (bR) are calculated by the following formulas:

$$b\mathrm{P} = \sum_{r=1}^{N} w_r \times P_r \tag{2.7}$$

$$b\mathrm{R} = \sum_{r=1}^{N} w_r \times R_r \tag{2.8}$$

where N is the number of references in the collection and w_r is the weight of the reference r in the collection. The value of each w_r is commonly defined as $1/N$.

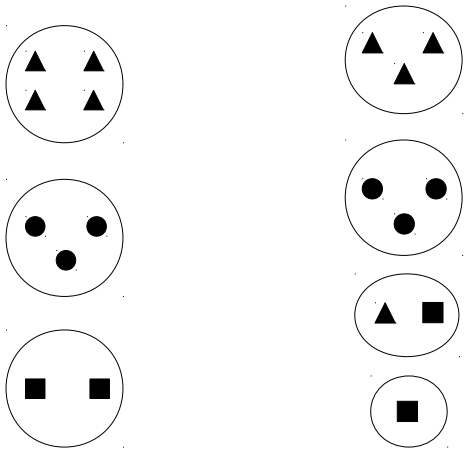
The harmonic mean (bF_α) of B-Cubed precision and recall is calculated by:

$$b\mathrm{F}_\alpha = \frac{1}{\alpha\frac{1}{bP} + (1-\alpha)\frac{1}{bR}} \tag{2.9}$$

Application of the metrics - an illustrative example

Consider the following example (see Figure 2.1): We have three theoretical clusters and four empirical clusters. Only one empirical cluster is not pure and there are two references fragmented into two clusters.

Table 2.2 shows the results of each metric applied to the illustrative example showed in Figure 2.1. We can notice that ACP and AAP of K metric and bP and bR of B-Cubed metrics produce similar results and that pF1 does not consider references



(a) Theoretical clusters (b) Empirical clusters

Figure 2.1. An illustrative example. Each geometric figure represents a reference to an author. The same figures refer to the same author.

Table 2.2. Performance of the evaluation metrics.

Metric	Result	
K	$\text{ACP} = \frac{1}{9} \times (\frac{3^2}{3} + \frac{3^2}{3} + \frac{1^2}{2} + \frac{1^2}{2} + \frac{1^2}{1}) = 0.89$	K = 0.81
	$\text{AAP} = \frac{1}{9} \times (\frac{3^2}{4} + \frac{3^2}{3} + \frac{1^2}{3} + \frac{1^2}{2} + \frac{1^2}{1}) = 0.73$	
pF1	$\text{pP} = \frac{3+3+0+0+0}{3+3+1+0} = 0.84$	pF1= 0.70
	$\text{pR} = \frac{3+3+0+0+0}{6+3+1} = 0.60$	
cF1	$\text{cP} = \frac{1}{4} = 0.25$	cF1= 0.28
	$\text{cR} = \frac{1}{3} = 0.33$	
RCS	$\text{RCS} = \frac{4}{3} = 1.33$	
B-Cubed	$\text{bP} = \frac{1}{9}(\frac{3}{3} + \frac{3}{3} + \frac{3}{3} + \frac{3}{3} + \frac{3}{3} + \frac{3}{3} + \frac{1}{2} + \frac{1}{2} + \frac{1}{1}) = 0.89$	$\text{bF}_{\alpha=0.5} = 0.68$
	$\text{bR} = \frac{1}{9}(\frac{3}{4} + \frac{3}{4} + \frac{3}{4} + \frac{3}{3} + \frac{3}{3} + \frac{3}{3} + \frac{1}{4} + \frac{1}{2} + \frac{1}{2}) = 0.72$	

which cannot be paired with other ones of the same author in the same empirical cluster.

2.4 Collections

Among the collections more commonly used to evaluate the author name disambiguation methods we can mention CiteSeer, DBLP, Penn, BDBComp and Rexa[1] that contain publications of computer science researchers, arXiv[2] that contains citations from high physics publications, BioBase[3] that contains citations from biological publications,

[1] http://rexa.info/

[2] http://www.cs.cornell.edu/projects/kddcup

[3] http://www.elsevier.com/wps/find/bibliographicdatabasedescription.cws_home/600715/description#description

IMDb[4] that contains data from movies, MEDLINE and BioMed that contain data from biomedical publications and Cora[5] that contains data on duplicate citations. In this section, we describe in more details DBLP, perhaps the most used of all previously mentioned collections [Han et al., 2004, 2005b,a; Pereira et al., 2009; Yang et al., 2008], and BDBComp, a collection built by us, that has the distinctive property that many authors possess only one publication, making the disambiguation task even harder. We exploit both collections in this thesis for evaluation purposes.

The collection of references extracted from DBLP sums up 4,287 references associated with 220 distinct authors, which means an average of approximately 20 references per author. This collection includes 2,270 references whose author names are in short format. Small variations of this collection have been used in several other works [Han et al., 2004, 2005b,a; Pereira et al., 2009; Yang et al., 2008]. Its original version was created by Han et al. [2004], and they manually labeled the references. 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 references for which they had insufficient information to be judged were eliminated. Han et al. [2004] also replaced the abbreviated publication venue titles by their complete version obtained from DBLP. We used 11 ambiguous groups extracted by Han et al. [2004] with some corrections.

The collection of references extracted from BDBComp sums up 361 references associated with 184 distinct authors, approximately two references per author, in which only eigth author names are in short format. Notice that, although much smaller than the DBLP collection, this collection is very difficult to disambiguate, because it has many authors with only one citation. This collection was created by us and contains the 10 largest ambiguous groups found in BDBComp at the time of its creation.

Table 2.3 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 citation.

As mentioned before, each reference has the author name, a list of coauthor names, the title of the work and the title of the publication venue (conference or journal) attributes.

Figure 2.2 shows the authorship distribution within each of two representative groups of each collection. Notice that, for a given group, few authors are very prolific

[4]http://www.imdb.com
[5]http://www.cs.umass.edu/ mccallum/code-data.html

Table 2.3. The DBLP and BDBComp collections

DBLP		BDBComp	
Ambiguous Group	#References/ #Authors	Ambiguous Group	#References/ #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	–	–

and appear in several citations, while most of the authors appear in only few citations (the same trend is observed in all groups of DBLP and BDBComp). This is an intrinsic characteristic of scientific publications, as pointed in [Liming and Lihua, 2005].

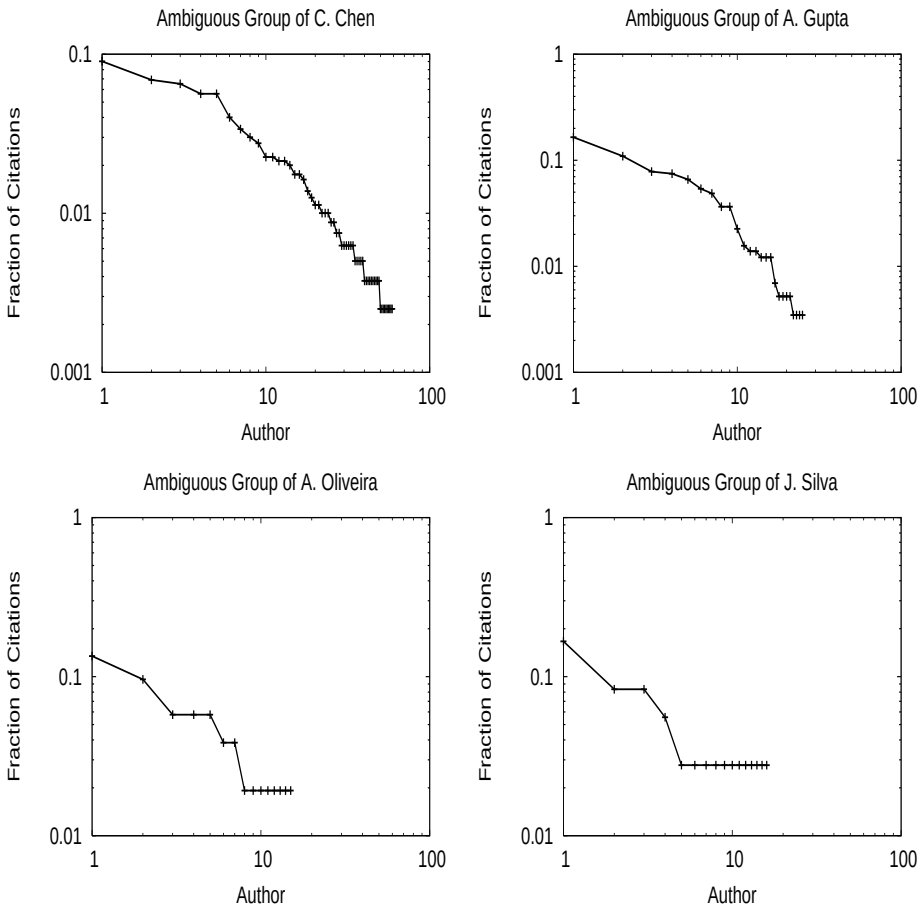


Figure 2.2. Authorship distribution within each ambiguous group. Authors (x-axis) are sorted in decreasing order of prolificness (i.e., more prolific authors appear in the first positions).

Chapter 3

Automatic Author Name Disambiguation Methods

In this chapter, we propose a taxonomy [Ferreira et al., 2012b] for characterizing the author name disambiguation methods in scholarly digital libraries and present an overview of representative author name disambiguation methods.

3.1 A Taxonomy for Author Name Disambiguation Methods

This section presents a hierarchical taxonomy for grouping the most representative automatic author name disambiguation methods found in the literature. The proposed taxonomy is shown in Figure 3.1. The methods may be classified according to the main type of exploited approach: *author grouping* [Bhattacharya and Getoor, 2007; Cota et al., 2010; Culotta et al., 2007; Fan et al., 2011; Ferreira et al., 2010; Han et al., 2005b; Huang et al., 2006; Kanani et al., 2007; Kang et al., 2009; On and Lee, 2007; Pereira et al., 2009; Soler, 2007; Song et al., 2007; Torvik et al., 2005; Torvik and Smalheiser, 2009; Treeratpituk and Giles, 2009; On et al., 2006; Yang et al., 2008], which tries to group the references to the same author using some type of similarity among reference attributes, or *author assignment* [Bhattacharya and Getoor, 2006; Ferreira et al., 2010; Han et al., 2004, 2005a; Tang et al., 2012], which aims at directly assigning the references to their respective authors. Alternatively, the methods may be grouped according to the evidence explored in the disambiguation task: the citation attributes (only), Web information, or implicit data that can be extracted from the available information.

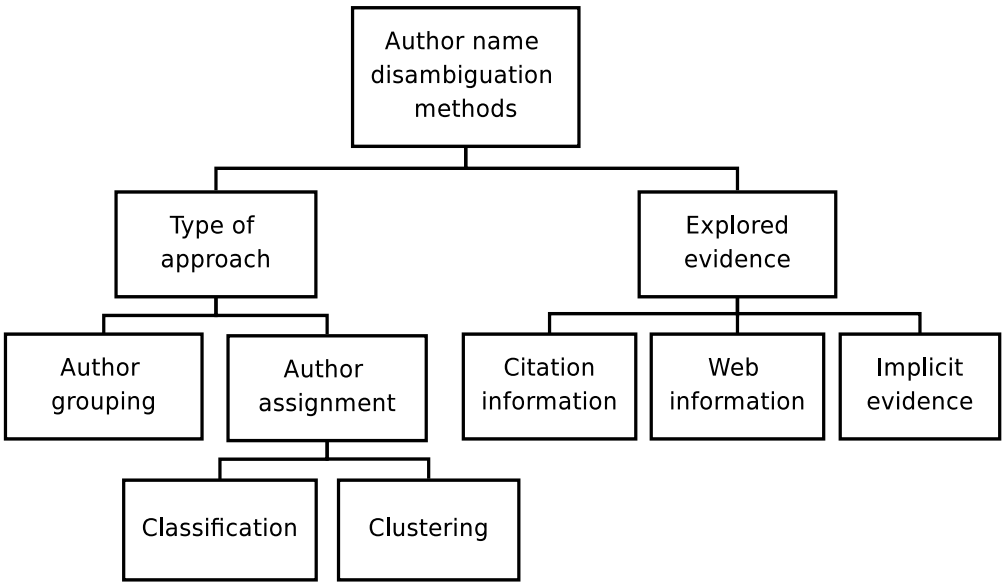


Figure 3.1. A taxonomy for author name disambiguation methods.

Notice that in this chapter we cover only automatic methods. Other types of method, such as manual assignment by librarians [Scoville et al., 2003] or collaborative efforts[1], rely heavily on human efforts, which prevent them from being used in massive name disambiguation tasks. For this reason, they are not addressed in this chapter. There are also efforts to establish a unique identification to each author, such as the use of an Open Researcher Contributor Identification[2] (ORCID), but these are also not covered here.

Since the name disambiguation problem is not restricted to a single context, it is also worth noticing that several other name disambiguation methods, which exploit distinct pieces of evidence or are targeted at other applications (i.e., name disambiguation in Web search results), have been described in the literature [Bekkerman and McCallum, 2005; Diehl et al., 2006; Galvez and de Moya Anegón, 2007; Vu et al., 2007; Yoshida et al., 2010]. However, a discussion of these methods is outside the scope of this chapter.

Finally, we should stress that the categories in our taxonomy are not completely disjoint. For instance, there are methods that use two or more types of evidence or mix approaches. In the next subsections, we detail our proposed taxonomy.

[1] http://meta.wikimedia.org/wiki/WikiAuthors
[2] http://www.orcid.org

3.1.1 Type of Approach

As said before, one way to organize the several existing author name disambiguation methods is according to the type of approach they exploit. We elaborate this distinction further in the discussion below.

3.1.1.1 Author Grouping Methods

Author grouping methods apply a similarity function to the attributes of the references (or group of references) in order to decide whether to group the corresponding references using a clustering technique. The similarity function may be predefined (based on existing ones and depending on the type of the attribute) [Bhattacharya and Getoor, 2007; Cota et al., 2010; Han et al., 2005b; On and Lee, 2007; Soler, 2007], learned using a supervised machine learning technique [Culotta et al., 2007; Huang et al., 2006; Torvik et al., 2005; Torvik and Smalheiser, 2009; Treeratpituk and Giles, 2009], or extracted from the relationships among authors and coauthors, usually represented as a graph [Fan et al., 2011; Levin and Heuser, 2010; On et al., 2006]. The defined similarity function is then used along with some clustering technique to group references of a same author, trying to maximize intra and minimize inter-cluster similarities, respectively.

Defining a Similarity Function

Here, a similarity function is responsible for determining how similar two references (or groups of references) to authors are. The goal is to obtain a function that returns high similarity values for references to the same author and returns low similarity values for references to different authors. Moreover, it is desirable that the similarity function be transitive. More specifically, let c_1, c_2 and c_3 be three citation records, if c_1 and c_2 are very similar (according to the function) and c_2 and c_3 are also very similar, then c_1 and c_3 should have high similarity according to our function. Next, we discuss the ways to determine this similarity function.

Using Predefined Functions

This class of methods has a specific predefined similarity function $\mathcal{S}$ embedded in their algorithms to check whether two references or groups of references refer to the same author. Examples of such function $\mathcal{S}$ include [Cohen et al., 2003]: the Levenshtein distance, Jaccard coefficient, cosine similarity, soft-TFIDF and others [Cohen et al., 2003], applied to elements of the reference attributes. Ad-hoc combinations of such functions have also been used (e.g., in [Bhattacharya and Getoor, 2007; Soler, 2007])

These methods do not need any type of supervision in terms of training data but their similarity functions are usually tuned to disambiguate a specific collection of citation records. For different collections, a new tuning procedure may be required. Finally, not all the functions used in these methods are transitive by nature.

Learning a Similarity Function

Learning a specific similarity function usually produces better results, since these learned functions are directly optimized for the disambiguation problem at hand. To learn the similarity function, the disambiguation methods receive a set $\{s_{ij}\}$ of pairs of references (the training data) along a special variable that informs whether these two corresponding references refer to the same author. The pair of references, r_i and $r_j \in R$ (the set of references) are usually represented by a similarity vector $\vec{s_{ij}}$. Each similarity vector $\vec{s_{ij}}$ is composed of a set $\mathcal{F}$ of q features $\{f_1, f_2, \ldots, f_q\}$. Each feature f_p of these vectors represents a comparison between attributes $r_i.A_l$ and $r_j.A_l$ of two references, r_i and r_j.

The value of each feature is usually defined using other functions, such as Levenshtein distance, Jaccard coefficient, Jaro-Winkler, cosine similarity, soft-TFIDF, euclidean distance, etc., or some specific heuristic, such as the number of terms or coauthor names in common, or special values such as the initial of the first name along with the last names, etc.

The training data is then used to produce a similarity function $\mathcal{S}$ from R x R to $\{0, 1\}$, where 1 means that the two references do refer to the same author and 0 means that they do not. As mentioned before, methods relying in learning techniques to define the similarity function are quite effective in different collections of citations, but they usually need many examples and sufficient features to work well, which can be very costly to obtain.

Exploiting Graph-based Similarity Functions

The methods that exploit graph-based similarity functions for author name disambiguation usually create a coauthorship graph $G = (V, E)$ for each ambiguous group. Each element of the author name and coauthor name attributes is represented by a vertex $v \in V$. The same coauthor names are usually represented by only a unique vertex. For each coauthorship (i.e., a pair of authors who publishes an article) an edge $\langle v_i, v_j \rangle \in E$ is created. The weight of each edge $\langle v_i, v_j \rangle$ is related to the amount of articles coauthored by the corresponding author names represented by vertices v_i and v_j.

A graph-based metric (e.g., shortest path as in [Levin and Heuser, 2010]) may be

combined with other similarity functions on the attributes of the references to authors or used as a new feature in the similarity vectors.

Clustering Techniques

Author grouping methods usually exploit a clustering technique in their disambiguation task. The most used techniques are partitioning, hierarchical agglomerative clustering, density-based and spectral clustering [Han and Kamber, 2005]. In general, these clustering techniques rely on a "good similarity function" to group the references. Next, we provide a brief description of these techniques applied to the author name ambiguity problem.

Partitioning Clustering Technique

A partitioning clustering technique, applied to the author name ambiguity problem, creates k partitions of the set of references to authors. These methods usually receive the number k of author groups to be created as input as well as the set of references to be disambiguated. They create an initial partitioning of k clusters (usually randomly) and, to improve the disambiguation process, move references to authors from one cluster to another based on some similarity criteria. The aim is that, in the end of the process, the references to a same author will be put together in the same cluster while references to different authors will remain in different clusters.

One advantage of these partitioning techniques is that a reference may be assigned to different authors during the disambiguation process, which can potentially help reducing erroneous assignments. This does not occur in hierarchical agglomerative clustering techniques (see below). However, these methods usually need to know the correct number of authors to perform well, which in most of cases is an unrealistic assumption. Moreover, similarities are usually calculated with respect to a representative reference within the clusters (e.g., a centroid). Thus, references that are not similar enough to this representative one but are similar to other references in the cluster may not be inserted into this (perhaps correct) cluster.

Hierarchical Agglomerative Clustering

A hierarchical agglomerative clustering technique [Han and Kamber, 2005] groups the references to authors in a hierarchical manner. Initially, each reference corresponds to a single cluster. Next, in each iteration of the process, the two most similar clusters are grouped together and the similarity among all clusters is recalculated. The process finishes when there is only a single cluster fusing all others or the similarity between the clusters reaches a given threshold.

One disadvantage of this technique is that if two references to different authors are put together in a same cluster during the process, they can no longer be moved to different clusters for the remainder of the process, i.e., this type of error cannot be corrected. In the case of the name disambiguation task, this particular homonym problem is one of the hardest to correct. An other disadvantage is the cost: we usually need to compare all clusters with each other to find the most suitable to be fused.

Density-based Clustering

With density-based clustering, a cluster corresponds to a dense region of references to authors surrounded by a region of low density (according to some density criteria). References in regions with low density are considered as noise.

An example of a density-based clustering algorithm that has been used in the author name disambiguation task is DBSCAN [Han and Kamber, 2005]. DBSCAN estimates the density of references by counting the number of references within a specified radius. DBSCAN classifies each reference as core references (i.e., references whose number of neighborhood references within a specific radius exceeds a given threshold), border references (i.e., a reference that is not a core reference but is within the neighborhood of a core reference) and noise references (i.e., a reference that is neither core nor border).

DBSCAN initially labels all references as core, border or noise based on the procedure described above. Next, it disconsiders all noise references and introduces edges between the core references whithin a given radius of each other. Each group of connected references is a cluster and each border reference is associated with one cluster of its core references.

One advantage of density-based clustering techniques is that the clusters are constructed using several representative references to authors. A disadvantage is that they are very sensible to their thresholds.

Spectral Clustering

Spectral clustering techniques [Zha et al., 2001] are graph-based techniques that compute the eigenvalues and eigenvectors, the spectral information, of a Laplacian Matrix that, in the the author name disambiguation task, represents a similarity matrix of a weighted graph $G = (V, E)$. In the name disambiguation task, each vertex $v \in V$ represents a reference to an author and each weighted edge $\langle v_i, v_j \rangle$ represents the similarity between the attributes of the vertices v_i and v_j. A graph-based technique splits the vertices into clusters by maximizing the weights of intra-cluster vertices and minimizing the weights of the inter-clusters vertices. A spectral clustering technique uses

the spectral information (i.e., eigenvalues and eigenvectors) instead of the similarity matrix in the clustering process.

Spectral clustering usually produces better performance than traditional clustering techniques. However, the spectral clustering method used in [Han et al., 2005b] for author name disambiguation needs to know the correct number of the authors (clusters) which, as discussed before, can be unrealistic in real scenarios.

3.1.1.2 Author Assignment Methods

Author assignment methods directly assign each reference to a given author by constructing a model that represents the author (for instance, the probabilities of an author publishing an article with other (co-)authors, in a given publication venue and using a list of specific terms in the work title) using either a supervised classification technique [Ferreira et al., 2010; Han et al., 2004] or a model-based clustering technique [Bhattacharya and Getoor, 2006; Han et al., 2005a].

Classification

Methods in this class assign the references to their authors using a supervised machine learning technique. More specifically, they receive as input a set of references to authors with their attributes called the *training data* (denoted as $\mathcal{D}$) that consists of examples or, in this case, references 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 $\{a_1, a_2, \ldots, a_n\}$, in which 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 references 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 references 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 no other) references in which the ith author is included.

These methods are usually very effective when faced with a large number of examples of citations for each author. Another advantage is that, if the collection has been disambiguated (manually or automatically), the methods may be applied only to references of the new citations inserted into the collection by simply running the learned model on them. Although successful cases of the application of these methods

have been reported, the acquisition of training examples usually requires skilled human annotators to manually label references. 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 examples for all possible authors are included in the training data and the authors change their interest area over time, new examples need be insert into training data continuously and the methods need to be retrained periodically in order to maintain their effectiveness.

Clustering

Clustering techniques [Han and Kamber, 2005] that attempt to directly assign references to authors work by optimizing the fit between a set of references to an author and some mathematical model used to represent that author. They use probabilistic techniques to determine the author in a iterative way to fit the model (or estimate the parameters in probabilist techniques) of the authors. For instance, in the first run of such a method each reference may be randomly distributed to an author a_i and a function, from a set of features $\{f_1, f_2, \ldots, f_m\}$ to $\{a_1, a_2, \ldots, a_n\}$, is derived using this distribution. In the second iteration, this function is used to predict the author of each reference and a new function is derived to be used in the next iteration. This process continues until a stop condition is reached, for instance, after a number of iterations. Two algorithms commonly used to fit the models in disambiguation tasks are Expectation-Maximization (EM) [Dempster et al., 1977] and Gibbs Sampling [Griffiths and Steyvers, 2004].

These methods do not need training examples, but they usually require privileged information about the correct number of authors or the number of author groups (i.e., group of authors that publish together) and may take some time to estimate their parameters (e.g., due to the several iterations). Additionally, these methods may be able to directly assign authors to their references in a new citations using the final derived function.

3.1.2 Explored Evidence

In this section, we describe the kinds of evidence most commonly explored by the disambiguation methods.

Citation Information

Citation information are the attributes directly extracted from the citations, such as author and coauthor names, work title, publication venue title, publication year, and so on. These attributes are the ones commonly found in all citations, but usually they are not sufficient to perfectly disambiguate all references to authors. Some methods also assume the availability of additional information, such as e-mail addresses, postal addresses, page headers etc., which are not always available or easy to obtain, although if existent, they usually help the process.

Web Information

Web information represents data retrieved from the Web that is used as additional information about an author publication profile. This information is usually obtained by submitting queries to search engines based on the values of citation attributes and the returned Web pages are used as new evidence (attributes) to calculate the similarity among references to authors. The new evidence usually improves the disambiguation task. One problem is the additional cost of extracting all the needed information from the Web documents.

Implicit Evidence

Implicit evidence is inferred from visible elements of attributes. Several techniques have been implemented to find implicit evidence, such as the latent topics of a citation. One example is the Latent Direchlet Location (LDA) [Blei et al., 2003] that estimates the topic distribution of a citation (i.e., LDA estimates the probability of each topic given a citation). This estimated distribution is used as new evidence (attribute) to calculate the similarity among references to authors.

3.2 Overview of Representative Methods

In this section, we present a brief overview of representative author name disambiguation methods which fall under one or more categories of the proposed taxonomy. Our main focus here is on those methods that have been specifically designed to address the name ambiguity problem in the context of bibliographic citations, since they are more related to the scope of this work. In the next subsections, we describe each method under the category we consider that best fits it. We notice that most of the described

methods explore citation information in the disambiguation task. Thus, we leave to Subsection 3.3 the discussion of those methods that use additional evidence.

Although not part of our taxonomy, one important point to understand the discussion that follows is the evaluation metrics that are used by each proposed method in their experimental evaluations. In addition to the metrics discussed in Section 2.3, some disambiguation methods also use *accuracy*, which is basically the proportion of correct results among all predictions, the traditional metrics of *precision*, *recall*, and *F1* [Rijsbergen, 1979], commonly used for information retrieval and classification problems[3] and MUC [Bagga and Baldwin, 1998]. In this last metric, recall is calculated by summing up the number of elements in the theoretical clusters minus the number of empirical clusters (obtained with the method) that contain these elements and dividing this by the total of elements minus the number of theoretical clusters. Precision is calculated similarly.

3.2.1 Author Grouping Methods

Using Predefined Functions

Han et al. [2005b] represent each reference as a feature vector where each feature corresponds to an element of a given instance of one of its attributes. The authors consider two options for defining the feature weights: TFIDF [Baeza-Yates and Ribeiro-Neto, 1999] and NTF (Normalized Term Frequency), being NTF given by $ntf(i,d) = freq(i,d)/maxfreq(i,d)$ where $freq(i,d)$ refers to the feature frequency i within the record d, and $maxfreq(i,d)$ refers to the maximum term frequency of feature i in the record d. The authors propose the use of K-way spectral clustering with QR decomposition [Zha et al., 2001] to construct clusters of references to the same author. To use this clustering technique, the correct number of clusters to be generated needs to be informed. The K-way spectral clustering method represents each reference as a vertex of an undirected graph and the weight of the edge between two vertices represents the similarity between the attributes associated with the respective references. K-way spectral clustering splits the graph so that records that are more similar to each other will belong to the same cluster. This method was evaluated using data obtained from the Web and DBLP. Experimental results achieved 63% of accuracy in DBLP and up to 84.3% in the Web collection.

An algorithm for collective entity resolution (i.e., an algorithm that uses only disambiguated coauthor names when disambiguating an author name of a citation) that

[3]In this last case, the authors are considered as classes and the correct assignments need to be known a priori.

exploits attribute elements (i.e., value of attributes present in the citation records) and relational information (i.e., authorship information between entities referred in the citations records) is proposed by Bhattacharya and Getoor [2007]. The authors propose a combined similarity function defined on attributes and relational information. As the initial step, the authors create clusters of disambiguated references verifying if two references have at least k coauthor names in common (they used only the author names in their experiments, but mention that other attributes may be used). The experiments were performed using soft-TFIDF, Jaro-Winkler, Jaro and Scaled Levenshtein measures for name attributes, and for relational attribute they used Common Neighbors, Jaccard coefficient, Adamic/Adar similarity and Higher-order neighborhood measures. The authors exploit a greedy agglomerative strategy that merges the most similar clusters in each step. The collections used in the experiments were a subset of CiteSeer containing machine learning documents, a collection of high energy physics publications from arXiv that was originally used in the KDD Cup 2003[4] and BioBase[5], containing biological publications of Elsevier and was used in an IBM KDD-Challenge competition. The method obtained around 0.99 of F1 in the CiteSeer and arXiv collections and around 0.81 in the BioBase collection.

Soler [2007] proposes a new distance metric between two citations, c_i and c_j, (or clusters of citations) based on the probability of these publications having terms and author names in common. In that work, the author proposes a semi-automatic algorithm that creates clusters of articles using the proposed metric and summarizes the clusters by means of a representative citation of the cluster including the distance from it to the others. Soler groups the citations for which the inter-citation distance is minimum using as evidences the author names, email, address, title, keywords, research field, journal and publication year attributes. The final decision on whether two candidate clusters belong to the same author or not is given by a specialist. He presents some illustrative cases of clusters obtained using his metric with records extracted from ISI-Thomson Web of Science database[6] but a more formal evaluation was not performed.

Cota et al. [2010] propose a heuristic-based hierarchical clustering method for author name disambiguation that involves two steps. In the first step, the method creates clusters of references with similar author names that share at least a similar coauthor name. Author name similarity is given by a specialized name comparison

[4]http://www.cs.cornell.edu/projects/kddcup

[5]http://www.elsevier.com/wps/find/bibliographicdatabasedescription.cws_home/600715/description#description

[6]http://isiknowledge.com

function called *Fragments*. This step produces very pure but fragmented clusters. Then, in the second step, the method successively fuses clusters of references with similar author names according to the similarity between the citation attributes (i..e., work title and publication venue) calculated using the cosine measure. In each round of fusion, the information of fused clusters is aggregated (i.e., all words in the titles are grouped together) providing more information for the next round. This process is successively repeated until no more fusions are possible according to a similarity threshold. The authors used pairwise F1 and K metrics on collections extracted from DBLP and BDBComp to evaluate the method and obtained around 0.77 and 0.93 for K in DBLP and BDBComp, respectively. An extension of this method that allows the name disambiguation task to be incrementally performed is presented in [Carvalho et al., 2011].

Learning a Similarity Function

Torvik et al. [2005] propose to learn a probabilistic metric for determining the similarity among MEDLINE records. The learning model is created using similarity vectors between two references. In that work, the similarity vector contains features resulting of the comparison between the normal citation attributes along with medical subject headings, language, and affiliation of two references. The authors also propose some heuristics for generating training sets (positive and negative) automatically. When the probabilistic metric receives the attributes associated with two references, their similarity vector is created and the relative frequency of this profile in the positive and negative training sets is checked for determining whether these two references refer to the same author or not. In a subsequent work, Torvik and Smalheiser [2009] extend this method by including additional features, new ways of automatically generating training sets, an improved algorithm for dealing with the transitivity problem and a new agglomerative clustering algorithm for grouping records. The authors estimate recall around 98.8%. They also estimate that only 0.5% of the clusters have mixed references of different authors (purity), and that only in 2% of the cases the references of the same author are split into two or more clusters (fragmentation).

Huang et al. [2006] present a framework for solving the name ambiguity problem in which a blocking method is first applied to create blocks of references to authors with similar names. Next DBSCAN, a density-based clustering method [Ester et al., 1996], is used for clustering references by author. For each block, the distance metric between pairs of citations used by DBSCAN is calculated by a trained online active support vector machine algorithm (LASVM), which yields, according to the authors, a simpler

and faster model than the standard support vector machines (SVMs). The authors use different functions for each different attribute, such as the edit distance for emails and URLs, Jaccard similarity for addresses and affiliations and soft-TFIDF for names. To demonstrate the effectiveness of this framework, the authors have applied it to a manually annotated dataset with 3,335 citation records and 490 distinct authors. Experiments were performed with pairs of references in which the disambiguator informs whether two references correspond to the same author or not. The authors obtained 0.906 in terms of pairwise F1. It should be noticed that these results were obtained by exploiting additional sources of evidence, such as the page headers of papers obtained from CiteSeer.

Culotta et al. [2007] aim to learn a score function to be applied to the disambiguation result, such that higher scores correspond to the more correct disambiguations. Instead of calculating the score using pairs of references, the authors propose a score function that considers all references in a cluster together, with the goal of maximizing the result of the score function in the resulting disambiguation. To learn this function, they propose a training algorithm that is error-driven, i.e., training examples are generated from incorrect predictions in the training data, and ranked, i.e., the classifier uses a ranking of candidate predictions to tune its parameters. The authors evaluated two loss functions to tune the parameters, Ranking Perceptron Freund and Schapire [1999] and Ranking MIRA Crammer and Singer [2003]. The experimental evaluation used two collections extracted from DBLP (one which is called Penn, because disambiguation was performed manually by students from Penn State University) and other from the Rexa[7] Digital Library. As evaluation metrics, they used pairwise F1, MUC and B-Cubed [Bagga and Baldwin, 1998]. As evidence, they exploited features such as first and middle names of the authors, number of coauthors in common, rarity of the last name, similarity between work titles, e-mails, affiliations and publication venue titles, as well as the minimum, maximum and average values for real-valued features, among several others. They also used a greedy agglomerative clustering technique to group the references. Ranking Perceptron generated the best results in DBLP and Penn, with 0.52 and 0.86 of pairwise F1, respectively. Ranking MIRA generates the best result on the other DBLP collection with 0.931 of pairwise F1.

Treeratpituk and Giles [2009] propose a learned similarity function for author name disambiguation in the MEDLINE digital library. The authors exploit a large feature set obtained from MEDLINE metadata, similar to that proposed in [Torvik et al., 2005]. The authors also use similarity vectors to learn the similarity function using a

[7] http://rexa.info

Random Forest classifier. They compare the use of Random Forests with decision trees, support vector machines, naïve Bayes and logistic regression to learn the function to be used along with some clustering technique (left unspecified). They also investigate the performance of subsets of the features capable of reaching good effectiveness. The authors obtain almost 96% of accuracy in their experiments by exploiting this large set of features.

Exploiting Graph-based Similarity Functions

On et al. [2006] address synonyms in the group entity resolution problem (i.e., a reference to a person associated with a group of items, e.g., an author with a list of publications) by proposing an approach that uses the quasi-clique graph-mining technique for exploiting, besides simple textual similarities, "contextual information" extracted from the group items' attributes (e.g., the citation attributes) as additional evidence. This contextual information is obtained constructing a graph for each group to represent relationships between the author names (i.e., references) and the attribute values (e.g., co-authors). This graph is then superimposed on the pre-built graph constructed using the entire set of author names. Using this contextual information, the authors also propose a graph-based distance function based on common quasi-clique between the graphs of two entities (i.e., references). They compared their graph-based function (distQC) with Jaccard, TF-IDF and IntelliClean functions [Lee et al., 2000] by measuring the precision and recall at the top k most similar references using three collections extracted from ACM[8], BioMed (a dataset of medical publications) and IMDb. On average, the experiments show an improvement of 63%, 83% and 46% over Jaccard, TFIDF and IntelliClean functions in terms of precision at top-k records returned by their algorithm in ACM. Similar results were obtained for the other collections.

Levin and Heuser [2010] propose a set of social network metrics that, together with string metrics, generate match functions (i.e., functions used to verify whether two references represent the same author). These functions were used in (very small) collections extracted from Cora[9], BDBComp and DBLP. The authors construct a graph with two kinds of vertices: one represents a reference to an author occurring in a citation and the other represents the citation itself; and two kinds of edges: one links the reference to the citation and the other links the vertices that share the same author name value. The authors obtained in their experiments around 95%, 82% and 95% of F1 in versions of Cora, BDBComp and DBLP, respectively.

[8]http://portal.acm.org
[9]http://www.cs.umass.edu/ mccallum/code-data.html

Fan et al. [2011] propose the GHOST (GrapHical framewOrk for name diSambiguaTion) framework. GHOST solves the homonym problem using only the coauthor name attribute in five steps. In the first one, GHOST represents a collection as a graph $G=(V, E)$, where each vertex $v \in V$ represents a reference to be disambiguated and each undirected edge $(v_i, v_j) \in E$ represents a coauthorship whose label S_{ij} is a set of citations coauthored by v_i and v_j. In the second step, GHOST identifies the valid paths eliminating the invalid ones between two nodes, i.e., a path that contains a subpath $v_i S_{ik} v_k S_{kj} v_j$ where S_{ik} is equal to S_{kj} and both have only one citation. In the third step, GHOST creates a matrix representing similarities between the vertices. For this, the authors propose a new similarity function based on the formula that calculates the resistance of a parallel circuit. In the fourth step, the Affinity Propagation clustering algorithm [Frey and Dueck, 2007] is used to group the references to the same author. Finally, in the last step, GHOST makes use of user feedback to improve the results. Experimental evaluation was performed in collections extracted from DBLP and MEDLINE. GHOST obtained on average 0.86 and 0.98 of pairwise F1 in DBLP and MEDLINE, respectively.

3.2.2 Author Assignment Methods

Classification

Han et al. [2004] propose two methods based on supervised learning techniques that use coauthor names, work titles and publication venues as evidence for assigning a reference to its author. The first method uses a naïve Bayes model (NB), a generative statistical model frequently used in word sense disambiguation tasks, to capture all writing patterns in the authors' citations. The second method is based on Support Vector Machines (SVMs), which are discriminative models basically used as a classifier [Mitchell, 1997]. An important difference between the two techniques is that a NB model requires only positive examples to learn about the writing patterns, whereas SVMs require both positive and negative examples to learn how to identify the author. Both methods have been evaluated with data taken from the Web and DBLP. Experimental results show that, on average, using all attributes, the SVM-based method was more accurate (accuracy=95.6%) than the NB method (accuracy=91.3%) for the Web collected dataset, while for the DBLP dataset the NB method performed better (SVM accuracy was 65.4% while NB's was 69.1%).

Veloso et al. [2012] propose SLAND, a disambiguation method that infers the author of a reference by using a supervised rule-based associative classifier. The proposed method uses author names, work title and publication venue title attributes as

features and infers the most probable author of a given reference r_i using the confidence of the association rules $\mathcal{X} \rightarrow a_i$ where $\mathcal{X}$ only contains features of r_i. The method also works on demand, i.e., the association rules to infer the correct author of a reference are generated in the moment of a disambiguation. The method is capable of inserting new examples into the training data during the disambiguation process, using reliable predictions, and detecting authors not present in the training data. Experiments were conducted in two collections extracted from DBLP and BDBComp and the proposed method outperformed representative supervised methods (e.g., SVM and NB) considering the Micro and Macro F1 metrics. In the DBLP and BDBComp collections, the (Micro) F1 values were 0.911 and 0.457, respectively. In order to deal with the cost of obtaining training data, this method was extended in [Ferreira et al., 2010] to become self-trained, i.e., it is now capable of producing its own training examples using (test) references to be disambiguated. Initially, the method extracts pure clusters of references by exploiting highly discriminative features, such as coauthor names. The most dissimilar clusters according to a given threshold are then selected to represent training examples for their authors. Next, the references in the rest of clusters are classified according to these training examples. In the experiments with the same collections, the self-trained method outperformed by far the unsupervised methods KWAY and SVM-DBSCAN and the associative method was the best choice for classifying the remaining test references not incorporated into the training data when compared to SVM and NB.

Clustering

Han et al. [2005a] present an unsupervised hierarchical version of the naïve Bayes-based method for modeling each author. In that work, the authors assume that each citation is generated by a mixture of K authors. They then calculate the probability of a citation record c_m given an author a_i, i.e., $\mathrm{P}(c_m|a_i)$ using the probability of each attribute of this record given such author, in a hierarchical way. To estimate the parameters, the authors use the Expectation Maximization algorithm [Dempster et al., 1977] aiming to maximize the likelihood of the citation records. The method obtained on average 54% and 58% of accuracy on data extracted from DBLP and the Web, respectively.

Bhattacharya and Getoor [2006] extend the generative model Latent Dirichlet Allocation (LDA) and propose a probabilistic model for collective entity resolution that uses the co-occurrence of the references to authors in each work to determine the entities jointly, i.e., they use the disambiguated references to disambiguate other references in the same citation. In their model, the authors associate an attribute v_a, that contains

the author name in the citation, with each author a. They assume that each citation is constructed by choosing their authors from an author group (i.e., a group of authors that publish some article together) distribution. That is, initially a distribution that determines the probability of each author group having a specific author chosen to write the article is selected. Next using this distribution, the authors and a variation of their names are chosen for this citation. The proposed method receives as input only an approximation of the number of author groups in the collection. Experiments were performed using citations extracted from CiteSeer and arXiv reaching up to 0.99 and 0.98 respectively of pairwise F1.

Tang et al. [2012] propose a probabilistic framework based on Hidden Markov Random Fields (HMRF) for the homonym subproblem. In this work, the authors use author names, work title, publication venue title, publication year, abstract and bibliographic references as content-based evidence and relationships between citations as structure-based evidence for disambiguating author names. Each relationship represents the fact that two citations were published in the same publication venue, have a coauthor name in common, cite the other, have distinct coauthor names that were coauthors in another citation, or have some specific user-provided constraint in common. Content and structure-based evidence are modeled as feature functions (used to represent the similarity between two citations by their content or relationships) which are then incorporated into a HMRF used to estimate the weights of the feature functions and to assign the citations to their authors. The authors also use Bayesian Information Criterion [Kass and Raftery, 1995] to estimate the number of authors of the collection. Experimental evaluation was performed on citations extracted from ArnetMiner[10]. Pairwise F1 values were 0.888 and 0.805 when the method uses the correct number of authors and when it estimates this number, respectively.

3.2.3 Using Additional Evidence

Web Information

Kanani et al. [2007] present two approaches for author name disambiguation that gather additional evidence from the Web. They construct a graph in which each vertex corresponds to a reference to an author and the edges are weighted with values that represent the probability of the two vertices (i.e., references) being the same author. This weight is initially calculated using the citation attributes. The authors propose two approaches to represent the information gathered from the Web. In the first, they

[10] http://arnetminer.org

use the result of searches submitted to a Web search engine for the work titles of citation records of the corresponding references to authors to change the weight of the edge between two references. In the second, they use one of the returned pages of the search as a new type of vertex in the graph (web vertex), adding new edges from this new vertex to each previously existing reference vertex, indicating the probability of the reference and the web page beloging to the same author. The proposed method learns a maximum entropy or logistic regression model for a pair of references a_i and a_j, and the weight of the edge $\langle a_i,a_j \rangle$ is given by the probability that the corresponding references refer to the same author minus the probability that these references refer to the different authors. In the end, a stochastic graph partitioning technique is used to cluster the references. DBLP, Penn and the Rexa collections were used in their experiments. Using the results of searches to Google to change the weight of the edges, their method obtains around 0.905, 0.877 and 0.918 of accuracy and around 0.886, 0.814 and 0.747 of pairwise F1 in the DBLP, Rexa and Penn collections, respectively. Experiments with the method that use the returned Web pages as vertices in the graph were run only with DBLP, producing 0.882 of accuracy and 0.903 of pairwise F1 in that collection.

Yang et al. [2008] address the author name ambiguity problem using topics and correlations found on the Web. They determine the topics of the citation from venue information using an extraction algorithm based on association rules in order to create a topic association network. They also use the Web for retrieving publication pages of authors or coauthors to be disambiguated. Then, they create a similarity function making use of an SVM classifier on top of all these features. The authors represent references to authors as vertices in a graph and the similarity function is used to create the edges between vertices. Their clustering technique removes a bridge edge when each resulting connected component has at least a given number of vertices. They tested their approach on the collection constructed by Han et al. [2004] and improved the accuracy by 66% (0.75 of accuracy) when compared to the use of citations without topics and Web correlations.

[Kang et al., 2009] exploit coauthorship information using a Web-based technique that obtains other (implicit) coauthors of the reference to be disambiguated. They submit a pair of author names of a same citation as a query to Web search engines to retrieve documents containing both author names and then extract new names found in these documents as new implicit coauthors of this pair. The authors measure the similarity between two references by counting the number of coauthors in common and use the single-link agglomerative clustering technique [Jain et al., 1999] to group the references to the same author. They used a collection of citations published in

Korean during 1999-2006 that has only the homonym problem, obtaining around 0.85 of pairwise F1.

Pereira et al. [2009] also exploit Web information to disambiguate author names. The proposed method attempts to find Web documents corresponding to curricula vitae or Web pages containing publications of a single author. It works in three steps. The first step receives a list of citations whose references must be disambiguated and, for each citation, submits a query containing data from its attributes to a Web search engine. It then inserts the top-m documents in the answer set into a set $\mathcal{D}$ of documents. The second step selects the documents in $\mathcal{D}$ that contain publication from a given author. The third step groups the reference to authors whose citations occur in a same document in a hierarchical manner, i.e., if citations of two ambiguous references occur in the same Web document, these citations are considered as belonging to the same author and are fused in a same cluster. The experimental evaluation was performed using data from DBLP, obtaining on average 0.80, 0.76 and 0.14 of K, pairwise F1 and cluster F1 metrics, respectively.

Implicit Evidence

Song et al. [2007] propose a two-step unsupervised method for author name disambiguation. The first step uses Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA) to assign a vector of probabilities of topics to each citation. The PLSA and LDA proposed by Song et al. introduce a variable for persons (authors) in the generative model, that does not exist in general generative models. The second step considers the distributions of the probability of topics with respect to citations as a new attribute for name disambiguation. The authors use the Levenshtein distance to measure the similarity between two names. When two names are considered similar, they use the probability vectors of two corresponding citations and the Euclidean distance to merge the citations of the same authors. The authors compared their method with a greedy agglomerative clustering, K-way spectral clustering and LASVM+DBSCAN on citations extracted from CiteSeer and personal names on the Web. Their experiments demonstrate that their method, when faced with a lot of citation information, is more effective than the baselines, obtaining on average around 0.911 and 0.936 of pairwise F1 on the Web and CiteSeer collections, respectively.

Shu et al. [2009] extend the Latent Dirichlet Allocation model (LDA) for obtaining the topic distribution of each citation by adding the assumption that every topic is a Dirichlet distribution over all author names, that each document is a mixture of topics, and that each topic is a Dirichlet distribution over all the words. They train

a classifier (C4.5 and SVMs) based on the similarity on topics, coauthor names, title and venue, as well as on the minimum distance between coauthor names, to predict whether two references correspond to the same author or not. The authors attempt to solve the problem of name ambiguity by trying to solve first the polysemy problem and then the synonymy. They use K-way spectral clustering to split the references into k sets, one for each author, in order to deal with the polysemy problem. Next, they compare two sets of references of authors whose names have a distance below a given threshold and count the number of citations from these two sets which are assigned to the same author by the classifier. This value is divided by the total number of pairs of those two sets and if the result is greater than a given threshold they are merged. The authors show the effectiveness of their method by applying it to data extracted from DBLP. For the polysemy problem the precision and recall were over 0.9 for the most ambiguous groups while for the synonym problem the precision was around 0.99 and recall was 0.917.

3.3 Summary of Characteristics

In this section, we present an overview of the characteristics found in the described author name disambiguation methods, which are summarized in Tables 3.1 and 3.2.

The collections used to evaluate the methods have been taken from: (1) CiteSeer, DBLP, BDBComp, ArnetMiner, and Rexa that contain publications of computer science researchers; (2) arXiv that contains citations from high energy physics publications; (3) BioBase, containing citations from biological publications; (4) MEDLINE and BioMed with data from biomedical publications; (5) ISI-Thomson with publications from several knowledge areas; (6) Cora, which consists of duplicated citations in Computer Science and person names extracted from the Web; and (7) IMDb with data about movie actors.

The majority of the described methods [Bhattacharya and Getoor, 2007; Cota et al., 2010; Culotta et al., 2007; Fan et al., 2011; Han et al., 2005b; Huang et al., 2006; Kanani et al., 2007; Kang et al., 2009; Levin and Heuser, 2010; On et al., 2006; Pereira et al., 2009; Shu et al., 2009; Soler, 2007; Song et al., 2007; Torvik and Smalheiser, 2009; Treeratpituk and Giles, 2009; Yang et al., 2008] try to disambiguate references to authors by using a similarity function to indicate whether two references refer to the same author instead of directly assigning the corresponding author to each reference, as proposed by some authors [Bhattacharya and Getoor, 2006; Ferreira et al., 2010; Han et al., 2004, 2005a; Tang et al., 2012; Veloso et al., 2012].

Table 3.1. Summary of characteristics - Author grouping methods

Method	Similarity function	Clustering technique	Evidence	Collections	Evaluation metric	Subproblem	# of authors
Bhattacharya and Getoor [2007]	Common neighbours, Jaccard, Adamic/Adar and Higher-order neighbourhoods	Agglomerative	Author name	CiteSeer, arXiv and BioBase	F1	Both	Unknown
Cota et al. [2010]	Fragment comparison and cosine	Agglomerative	Citation attributes	DBLP and BDBComp	Pairwise F1 F1 and K	Both	Unknown
Culotta et al. Culotta et al. [2007]	Error-drive and hank-based learning	Agglomerative	All of each collection	DBDL and Rexa	Pairwise F1, MUC and B-Cubed	Both	Unknown
Fan et al. [2011]	graph-based	Affinity Propagation	Author names	DBLP and MEDLINE	Pairwise F1	Homonym	Unknown
Han et al. [2005b]	Cosine	Spectral clustering	Citation attributes	DBLP and Web	Accuracy	Both	Known
Huang et al. [2006]	Learned using LASVM	DBScan	First page of the articles	CiteSeer	Pairwise F1	Both	Unknown
Kanani et al. [2007]	Learned using maximum entropy or logistic regression	Partitioning	Citation attributes and Web pages	DBLP, Penn and Rexa	Accuracy and pairwise F1	Both	Unknown
Kang et al. [2009]	Heuristic	Agglomerative	Author names and Web pages	Korean citations	F1 and under/ over-clustering error	Homonym	Unknown
Levin and Heuser [2010]	Social network metrics	-	Citation attributes	DBLP, Cora and BDBComp	F1	Both	Unknown
On et al. [2006]	Quasi-clique	-	Citation/Movie attributes	ACM, BioMed and IMDb	Ranked recall and precision	Synonym	Unknown
Pereira et al. [2009]	Heuristic	Agglomerative	Citation attributes	DBLP	Pairwise and cluster F1 and K	Both	Unknown
Shu et al. [2009]	Learned using C4.5/SVMs and edit distance	Spectral and agglomerative clustering	Citation attributes	DBLP	Pairwise F1	Both	Known
Soler [2007]	Probabilistic metric	Agglomerative	Citation attributes, email, address, keywords and research field	ISI-Thomson	-	Both	Unknown
Song et al. [2007]	Levenshtein and Euclidean distance	Agglomerative	Citation attributes and latent topics (LDA/PLSA)	CiteSeer and Web	Pairwise and cluster F1	Both	Unknown
Torvik and Smalheiser [2009]	Learn a probabilist metric	Agglomerative	MEDLINE metadata	MEDLINE	Recall	Both	Unknown
Treeratpituk and Giles [2009]	Learned using random forest classifier	-	MEDLINE metadata	MEDLINE	Accuracy	Both	Unknown
Yang et al. [2008]	Learned using SVM	Partitioning	Citation attributes, topics and Web pages	DBLP	Accuracy, precision and recall	Both	Unknown

Table 3.2. Summary of characteristics - Author assignment methods

	Method	Technique	Attributes	Collections	Evaluation metric	Subproblem	# of authors
Classification	Ferreira et al. [2010]	Associative classifier	Citation attributes	DBLP and BDBComp	Pairwise F1 and K	Both	Estimated
	Han et al. [2004]	SVM and naïve Bayes classifiers	Citation attributes	DBLP and Web	Accuracy	Both	Known
	Veloso et al. [2012]	Associative classifier	Citation attributes	DBLP and BDBComp	F1	Both	Estimated
Clustering	Bhattacharya and Getoor [2006]	LDA with Gibbs sampling	Author names	CiteSeer and arXiv	F1	Both	Estimated
	Han et al. [2005a]	Hierarchical naïve Bayes with EM	Citation attributes	DBLP and Web	Accuracy	Both	Known
	Tang et al. [2012]	Hidden Markov Random Fields	Citation attributes	ArnetMiner	Pairwise F1	Homonym	Estimated

Some of these methods receive the correct number of authors in the collection as input [Fan et al., 2011; Han et al., 2005a,b] or this number corresponds to the number of authors in the training data [Han et al., 2004]. Other methods, such as those proposed in [Bhattacharya and Getoor, 2006], [Tang et al., 2012] and [Ferreira et al., 2010], try to estimate this number.

Almost half of the proposed methods [Bhattacharya and Getoor, 2006, 2007; Cota et al., 2010; Fan et al., 2011; Ferreira et al., 2010; Han et al., 2004, 2005a,b; Levin and Heuser, 2010; On et al., 2006; Shu et al., 2009] use at most the three main citation attributes, namely, author names, work title and publication venue title, as disambiguation evidence. These attributes are the most commonly found in citation records, constituting in most cases the hardest situation for disambiguation. Few methods [Kanani et al., 2007; Kang et al., 2009; Pereira et al., 2009; Yang et al., 2008] exploit additional evidence such as emails, addresses, paper headers etc., which are not always available or easy to obtain.

Finally, Tables 3.1 and 3.2 also show the evaluation metrics used by each of the proposed methods as well as the type of subproblem (i.e., synonymy, homonym, or both) they tackled.

Chapter 4

SAND: Self-training Author Name Disambiguator

In this chapter, we describe our proposed hybrid disambiguation method, SAND (standing for *Self-training Author Name Disambiguator*) [Ferreira et al., 2010], which is one of the major contributions of this thesis. SAND exploits the strengths of both author grouping and author assignment methods. Specifically, it works in three steps. In the first step, *author grouping*, recurring patterns in the coauthorship graph are exploited in order to produce very pure clusters of references. In the second step, *cluster selection*, a subset of the clusters produced in the previous step is selected as training data for the next step. Then, in the third step, *author assignment*, a learned function is derived to disambiguate the references in the clusters that were not selected in the previous step. The final result, as we shall see, is a highly effective and extremely practical disambiguator. Experimental results, using references extracted from DBLP and BDBComp, as well as synthetic data produced with SyGAR, show that SAND outperforms all author grouping methods including state-of-the art ones and has competitive, sometimes superior, performance when compared with author assignment methods, without the need for any manually labeled data as required by those methods.

4.1 SAND Design

In the following sections, we will present a detailed description of the SAND steps. These steps are applied after a well-known pre-processing procedure, which includes blocking, stop-word removal, and stemming. Stop-word removal and stemming are performed on the words that compose work and publication venue titles. Moreover, authors with similar ambiguous names are grouped together (i.e., blocked), creating

ambiguous groups. Disambiguation operations are performed within each ambiguous group, so that useless comparisons involving non-ambiguous authors are avoided.

4.1.1 The Author Grouping Step

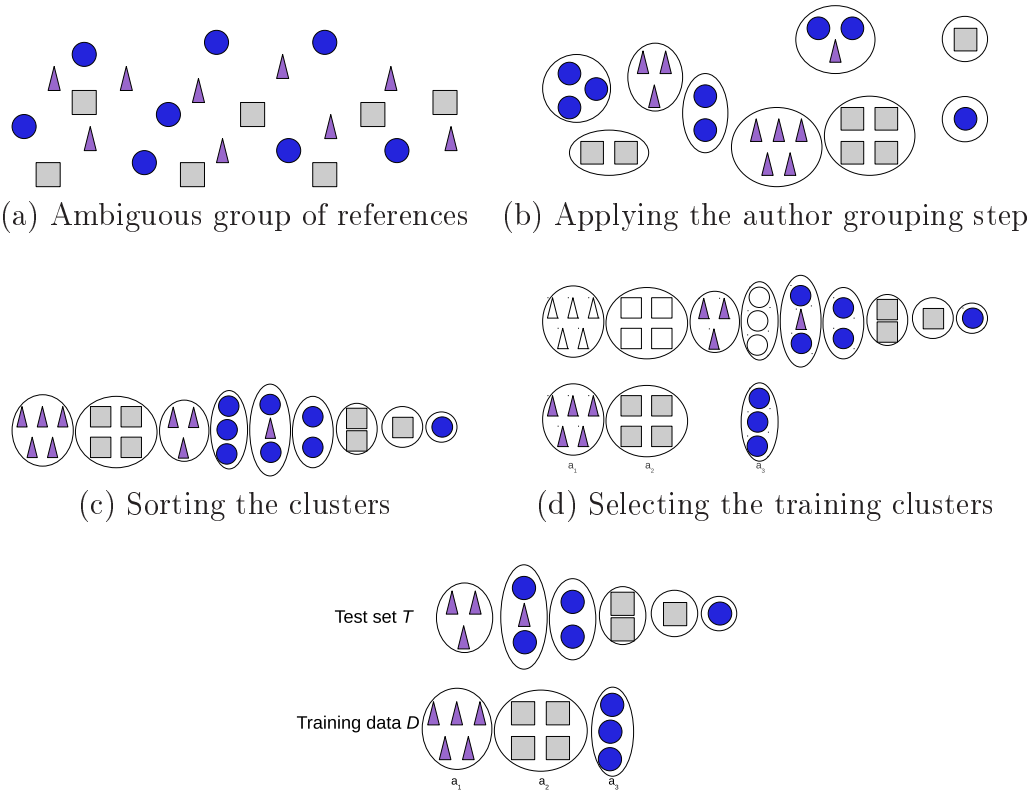
The goal of this step is to automatically create pure clusters of references. Some of these clusters will be selected by the cluster selection step to compose the training data to be used in the final step. The approach we adopt is to organize references within each ambiguous group into individual clusters, so that references placed in a same cluster tend to be very similar to each other and dissimilar to references placed in other clusters. The key intuition is that some of these clusters can be associated with a unique author label, therefore references within such cluster can serve as training examples.

In order to properly produce training examples, the extracted clusters should be as pure as possible, in the sense that each cluster should contain only references to one author. Otherwise, if a cluster with a low degree of purity (i.e., a cluster with references to distinct authors) is selected as training, then references to different authors could be assigned to the author label associated with this cluster in the author assignment step, increasing the homonym problem.

A straightforward way of extracting pure clusters is to ensure that each one of them contains only a single reference. In this case, clusters are totally pure, however, fragmentation is maximum, i.e., the references of a same author are placed into different clusters. Fragmented clusters are potentially detrimental for learning the author assignment function, since references to the same author would receive different author labels.

Accordingly, in SAND, pure clusters are extracted by exploiting highly discriminative attributes, so that references associated with different authors are unlikely to be grouped together into the same cluster. In the context of bibliographic citation, we have based this strategy on a general heuristic that assumes that very rarely two authors with similar names that have coauthors in common would be two different people in the real world [Cota et al., 2010].

Figure 4.1(a) and (b) illustrate the author grouping step. Each geometric figure represents a reference and the figures with the same shape are references to the same author. Algorithm 1 describes the author grouping step in details. This algorithm receives as input an ambiguous group of references G (see Figure 4.1(a)) and returns a list $\mathcal{C}$ of clusters of references (see Figure 4.1(b)). It processes G by splitting it into two separate lists: S with references whose author names occur in a short format



(a) Ambiguous group of references (b) Applying the author grouping step

(c) Sorting the clusters (d) Selecting the training clusters

(e) Training and test sets

Figure 4.1. Illustrative example. The author grouping and cluster selection steps.

(i.e., names with only the initial of the first name and the last name) and L with the remaining ones (i.e., those references whose ambiguous author names are not in short format). Then, it proceeds by first processing L (line 6) and then S (line 7). When processing the lists L and S, the initial clusters of references are built using the author name and the list of coauthor names as evidence. The idea of first processing the list of long names is that these names provide more reliable evidence for our similarity functions.

Algorithm 2 describes the function ProcessList used to process the list of references in Algorithm 1 (lines 6 and 7). This algorithm receives a list L of references and a list C_i of clusters of references and returns a new list C_o with each reference r from L in some cluster c of C_o. It compares the author name of each reference r with the author name of each cluster c[1] using some similarity function. If the author name of r is similar to the author name of c and there are coauthor names in r that are similar

[1] We use as the author name of a cluster that of the first reference inserted into the cluster. Remind that we process the long names first.

Algorithm 1 The Author Grouping Step

Input: Ambiguous group G of references;
Output: List $\mathcal{C}$ of clusters of references;

1: Let L and S be lists of references;
2: Let C_1 and C_2 be lists of clusters;
3: $S \leftarrow$ GetShortNameRecords(G);
4: $L \leftarrow$ GetLongNameRecords(G);
5: $C_1 \leftarrow \emptyset$;
6: $C_2 \leftarrow$ ProcessList(L,C_1);
7: $\mathcal{C} \leftarrow$ ProcessList(S,C_2);

Algorithm 2 Function ProcessList

Input: List L of references;
Input: List C_i of clusters of references;
Output: List C_o of clusters of references;

1: $C_o \leftarrow C_i$;
2: **for each** r in L **do**
3: $inserted \leftarrow$ false;
4: $c \leftarrow$ first(C_o);
5: **while not** $inserted$ **and** $c \neq$ **null do**
6: **if** similar(r.authorName, c.authorName) **and** exists similar(r.coauthorNames, c.coauthorNames) **then**
7: InsertReference(r, c);
8: $inserted \leftarrow$true;
9: **end if**
10: $c \leftarrow$ next(C_o);
11: **end while**
12: **if** $inserted$ = **false then**
13: $c \leftarrow$ CreateNewCluster(r);
14: Append(C_o, c);
15: **end if**
16: **end for**

to some coauthor names in c[2], r is inserted into this cluster c (line 7); otherwise a new cluster is created with this reference r (line 13).

To measure the similarity between two names we use a function derived from the Fragment Comparison algorithm, an edit-distance matching algorithm specially designed for persons' names [Oliveira, 2005]. To verify whether a reference r and a cluster c share coauthors we exploit two strategies, a weaker and a stronger one. The weaker strategy considers that r and c share coauthors when they have at least one

[2]We consider the set of all coauthor names of all references in a cluster as the value of the coauthor names attribute of such cluster.

similar coauthor name in common. The second, stronger strategy tries to increase the purity of the generated clusters, by building upon the first strategy. For this, we use a external source of evidence containing the most popular last names of a given language. In this second strategy, we consider that r and c share coauthors if both have at least one similar coauthor whose last name is not popular or if they have at least *two* similar coauthor names (popular or not). We use a list of popular last names extracted from Wikipedia[3] and from the BDBComp digital library (for the Brazilian Portuguese case) to compose our list of popular last names.

Though simple, this additional constraint tends to extract even purer clusters when compared to the first strategy, as it will be shown in our experiments. Unfortunately, both strategies also tend to fragment references to an author into multiple clusters. This is expected, since some authors are likely to have many different coauthors due to multiple interests and some of these coauthors may have never published together.

4.1.2 The Cluster Selection Step

As mentioned before, if the final set of clusters to be used as training data is too fragmented, then possibly many references will be associated with incorrect author labels[4], decreasing the benefit of the training examples. One strategy to reduce fragmentation in the training data is to select only the clusters belonging to different real authors.

Algorithm 3 describes the cluster selection step in details while Figure 4.1(c), (d) and (e) illustrate this step. The process of selecting the clusters whose references will compose the initial training data, starts by sorting the input clusters produced in the previous step (line 3) in descending order of size (i.e., the number of references within the cluster). The result is a sorted list $\mathcal{C}$ of clusters (see Figure 4.1(c)). Next, the largest cluster in $\mathcal{C}$ is inserted into the set of selected clusters, $\mathcal{S}$ (lines 4 and 5). This selected cluster is also removed from $\mathcal{C}$. As the clusters in $\mathcal{S}$ should belong to different authors, the next cluster in $\mathcal{C}$ to be inserted into $\mathcal{S}$ should be one not similar to any of 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 most likely to contain references associated with authors not already in $\mathcal{S}$. So, we insert a cluster $c_i \in \mathcal{C}$ in $\mathcal{S}$ if $\forall c_j \in \mathcal{S}$, c_i is not similar to c_j (lines from 6 to 11). The iteration continues with the next candidate cluster in $\mathcal{C}$. The process finally stops when the last cluster in $\mathcal{C}$ is evaluated (see

[3] *http://en.wikipedia.org/wiki/Lists_of_most_common_surnames*

[4] Remind that each cluster in the training data is associated with a different label. If two clusters of the same author are included in the training data, these clusters will be considered as belonging to different authors.

Algorithm 3 The Cluster Selecting Step

Input: List $\mathcal{C}$ of clusters of references;
Output: List $\mathcal{D}$ of training data;
Output: List $\mathcal{T}$ of test set;

1: Let $\mathcal{S}$ be the list of selected clusters;
2: $\mathcal{S} \leftarrow \emptyset$;
3: $\mathcal{C} \leftarrow$ Sort($\mathcal{C}$, desc);
4: $c_i \leftarrow$ GetFirstCluster($\mathcal{C}$);
5: Append($\mathcal{S}$, c_i);
6: Remove(c_i, $\mathcal{C}$);
7: **for each** c_i **in** $\mathcal{C}$ **do**
8: **if** $\forall c_j \in \mathcal{S}$, Dissimilar($c_i$, c_j) **then**
9: Append($\mathcal{S}$, c_i);
10: Remove(c_i, $\mathcal{C}$);
11: **end if**
12: **end for**
13: $\mathcal{D} \leftarrow \mathcal{S}$;
14: $\mathcal{T} \leftarrow \mathcal{C}$;

Figure 4.1(d)). At the end of the process, references in each cluster $c_j \in \mathcal{S}$ are inserted into the training data $\mathcal{D}$. Each reference receives the author label of the corresponding cluster. The remaining clusters whose references were not selected as training data, will compose the test set $\mathcal{T}$, which will be disambiguated by the last step of SAND (see Figure 4.1(e)).

We evaluate three strategies to measure the similarity/dissimilarity among clusters:

- *Strategy 1.* We compare two clusters c_i and c_j using the attributes of the references in these clusters. Each reference is represented as a feature vector and a similarity function ϕ (e.g., cosine, euclidean distance, etc.) between references in clusters c_i and c_j (or between their respective centroids), is used to measure the similarity between c_i and c_j. The clusters are considered dissimilar according to the following rule:

$$Dissimilar(c_i, c_j) = \begin{cases} 1, & \text{IF } \phi(c_i, c_j) < \phi_{min} \\ 0, & \text{OTHERWISE} \end{cases}$$

In other words, clusters c_i and $c_j \in \mathcal{S}$ are considered not similar if the value $\phi(c_i, c_j)$ between c_i and c_j is not greater than a minimum value (ϕ_{min}) necessary for the clusters to be considered similar.

- *Strategy 2.* We compare two clusters c_i and c_j using only the author name assigned to them, using some author name similarity function τ (e.g., fragment comparison) that checks whether two author names are similar (i.e., if they may refer to the same person). If the cluster's author names are considered to be not similar according to function τ, the respective clusters are also considered as dissimilar.

$$Dissimilar(c_i, c_j) = \begin{cases} 1, & \text{IF NOT } \tau(c_i.authorName, c_j.authorName) \\ 0, & \text{OTHERWISE} \end{cases}$$

- *Strategy 3.* This strategy combines both previous strategies.

$$Dissimilar(c_i, c_j) = \begin{cases} 1, & \text{IF NOT } \tau(c_i.authorName, c_j.authorName) \\ & \text{or } \phi(c_i, c_j) < \phi_{min} \\ 0, & \text{OTHERWISE} \end{cases}$$

As options for the function ϕ, we currently exploit the cosine similarity function and the euclidean distance that are metrics frequently used to measure the similarity or dissimilarity between vectors. For cluster similarity we used four options: similarity between the respective cluster centroids as well as single, complete and average linkage (described next). This encompasses eight possible combinations of similarity function and cluster similarity strategies. Next, we describe the similarity metrics in more detail, in which each reference r is represented as a feature vector $\vec{r}$.

Cosine

The cosine similarity function [Salton et al., 1975] is obtained from the following formula:

$$cosine(\vec{r_i}, \vec{r_j}) = \frac{\sum_k r_{ik} \cdot r_{jk}}{|\vec{r_i}| \cdot |\vec{r_j}|}$$

where,

- $\vec{r_i}$ and $\vec{r_j}$ correspond to the feature vectors of references r_i and r_j, respectively;
- $|\vec{r}|$ corresponds to the norm of the vector $\vec{r}$; and

- r_{ik} and r_{jk} correspond to the value of k-th feature in the vectors $\vec{r_i}$ and $\vec{r_j}$, respectively.

Euclidean Distance

The euclidian distance [Jain et al., 1999] between two vectors is calculated by the following formula:

$$euclidean\_distance(\vec{r_i}, \vec{r_j}) = \sqrt{\sum_{k=1}^{n}(r_{ik} - r_{jk})^2}$$

We change the euclidean distance to use it as a similarity metric by applying the following formula:

$$euclidean(\vec{r_i}, \vec{r_j}) = 1 - \frac{euclidean\_distance(\vec{r_i}, \vec{r_j})}{euclian\_distance_{max}}$$

where $euclidean\_distance_{max}$ corresponds to the largest distance between all vectors.

Cluster Similarity Strategies

In our experiments, we evaluate similarity strategies specially designed for clusters [Jain et al., 1999] based on (1) the centroids of the clusters, (2) single-link, (3) complete-link and (4) average-link. The similarities between two clusters c_i and c_j are calculated by using the following formulas:

- Centroid.

  $$centroid(c_i, c_j) = \phi(\vec{r_i}, \vec{r_j})$$

  where, $\vec{r_i}$ and $\vec{r_j}$ are the centroids from c_i and c_j, respectively, and $\vec{r_i} = \frac{1}{|c_i|}\sum_{r \in c_i}(\vec{r})$.

- Single-link.

  $$single(c_i, c_j) = \phi(\vec{r_i}, \vec{r_j})$$

  where, $\vec{r_i}$ and $\vec{r_j}$ are the vectors from c_i and c_j, respectively, that have the highest similarity.

- Complete-link.

  $$complete(c_i, c_j) = \phi(\vec{r_i}, \vec{r_j})$$

  where, $\vec{r_i}$ and $\vec{r_j}$ are the vectors from c_i and c_j, respectively, that have the lowest similarity.

- Average-link.

$$average(c_i, c_j) = \frac{\sum_{\vec{r_i} \in c_i} \sum_{\vec{r_j} \in c_j} \phi(\vec{r_i}, \vec{r_j})}{|c_i| * |c_j|}$$

$\phi(\vec{r_i}, \vec{r_j})$ can be calculated by using any similarity metric between two vectors.

4.1.3 The Author Assignment Step

In the third and final step of SAND, the set of examples, $\mathcal{D}$, is used to produce a disambiguation 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 of the references in the test set $\mathcal{T}$. In case of SAND, this test set is composed of all references not belonging to clusters selected in the previous step. The idea is that, with the training set selected in the previous step, we would be able to learn an assignment function that will correctly predict the authors of these remaining references. For those authors in the collection without a representative cluster in the training data $\mathcal{D}$, our method will (hopefully) detect them as new authors and include them in the training for later use, i.e., the method is also self-trained. Next, we describe the author assignment step, which is based on a lazy associative classifier [Veloso et al., 2006b] to produce disambiguation functions from $\mathcal{D}$.

Associative Name Disambiguation

The proposed technique for deriving a disambiguation function 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 technique 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 [Veloso et al., 2006b]. Typically, these associations are expressed using rules[5] 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 a reference x if $\mathcal{X} \subseteq x$ (i.e., x contains all features in $\mathcal{X}$) and this rule is included in $\mathcal{R}_{a_i}^x$. That is, $\mathcal{R}_{a_i}^x$ is composed of rules predicting author a_i and matching reference x. Obviously, $\mathcal{R}_{a_i}^x \subseteq \mathcal{R}_{a_i} \subseteq \mathcal{R}$.

[5] These rules can be efficiently extracted from $\mathcal{D}$ using the strategy proposed in [Veloso et al., 2006b].

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 [Veloso et al., 2006a], at disambiguation time. The method projects the search space for rules according to information in references in $\mathcal{T}$, allowing for efficient rule extraction. In other words, the proposed method projects/filters the training data according to the features in reference $x \in \mathcal{T}$, and extracts rules from this projected training data, which is denoted as $\mathcal{D}^x$. This ensures that only rules that carry information about reference x are extracted from the training data, drastically limiting the number of possible rules. The lines 1 to 5 of Algorithm 4 describes the projection.

Algorithm 4 Associative Name Disambiguation.

Input: Examples in $\mathcal{D}$ and reference $x \in \mathcal{T}$
Output: The predicted author of the reference x

1: Let $\mathcal{L}(f_i)$ be the set of examples in $\mathcal{D}$ in which feature f_i has occurred
2: $\mathcal{D}^x \Leftarrow \emptyset$
3: **for each** feature $f_i \in x$ **do**
4: $\quad \mathcal{D}^x \Leftarrow \mathcal{D}^x \cup \mathcal{L}(f_i)$
5: **end for**
6: **for each** author a_i **do**
7: $\quad \mathcal{R}^x_{a_i} \Leftarrow$rules $\mathcal{X} \rightarrow a_i$ extracted from $\mathcal{D}^x$
8: $\quad$ Estimate $\hat{p}(a_i|x)$, according to Equation 4.2
9: **end for**
10: Predict author a_i such that $\hat{p}(a_i|c) > \hat{p}(a_j|c) \forall j \neq i$

Predicting the Author of each Reference

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 [Agrawal et al., 1993] (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 the reference 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 the reference 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 reference 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 4.1 (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).

$$s(a_i, x) = \frac{\sum_{j=1}^{|\mathcal{R}^x_{a_i}|} \theta(r_j)}{|\mathcal{R}^x_{a_i}|} \tag{4.1}$$

The estimated probability of a_i being the author of reference x, denoted as $\hat{p}(a_i|x)$, is simply obtained by normalizing $s(a_i, x)$, as shown in Equation 4.2. 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 reference x. The lines 6 to 10 of Algorithm 4 describes the prediction of the author of each reference.

$$\hat{p}(a_i|x) = \frac{s(a_i, x)}{\sum_{j=1}^{n} s(a_j, x)} \tag{4.2}$$

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 reference $x \in \mathcal{T}$, 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 4.3.

$$\Delta(x) = \frac{\hat{p}(a_i|x)}{\hat{p}(a_j|x)} \tag{4.3}$$

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 [Geisser, 1993], 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 references in a queue, so that references associated with reliable predictions are considered first. The process works as follows. Initially, references in the test set are randomly placed in the queue. If the author of the reference that is located in the beginning of the queue can be reliably predicted, then the prediction is performed, the reference is removed from the queue and included into $\mathcal{D}$ as a new example. Otherwise, if the prediction is not reliable, the corresponding reference is simply placed in the end of the queue and will be only processed after all other references. The process continues performing more reliable predictions first, until no more reliable predictions are possible. The remaining references 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 references in $\mathcal{T}$ are processed.

Detecting New Authors

We propose to use the lack of rules supporting any already seen author (i.e., authors that are present in some reference 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 a reference $x \in \mathcal{T}$, 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 in $\mathcal{D}$.

Predicting the Author of each Cluster

Alternatively, instead of predicting the author of each reference, we could explore some of the work already done in the author grouping step in order to directly predict the author of the cluster, i.e., all references in a cluster $c \in \mathcal{T}$ would be assigned to the same author label avoiding to assign some of the references within the cluster to other

authors. This can only be done in an effective way if most of the clusters are pure, because mixed references in a cluster could not be fixed later.

To predict the author of a cluster c, we first predict the authors of each reference $r \in c$. After that, we use the parameter Δ_{min} and the number of references of the two most oftenly predicted authors, a_i and a_j, in c. Let these numbers be n_i and n_j, respectively. If $\frac{n_i}{n_j} > \Delta_{min}$, we consider the cluster c as belonging to author a_i and each reference $r \in c$ is assigned to author a_i. Otherwise, we assign the references in c to a new author. After the prediction of the cluster c, its references are inserted into the training data $\mathcal{D}$

4.2 Experimental Evaluation

In this section we present experimental results that demonstrate the effectiveness of SAND. In order to evaluate the effectiveness of our disambiguation method, we used collections of references extracted from DBLP and BDBComp (see Section 2.4) as well as synthetic data produced with SyGAR, our generator of synthetic citation records that is described in Chapter 5. We also use the K and pairwise F1 metrics (see Section 2.3) in this evaluation.

We compare the effectiveness of SAND against six baselines: three unsupervised author grouping and three supervised author assignment methods. The three author assignment methods are the ones proposed by Han et al. [2004] and the state-of-the-art method proposed by Veloso et al. [2012]. The first method, referred to as NB, uses the naïve Bayes probability model [Mitchell, 1997], the second one, referred to as SVM, uses Support Vector Machines (SVMs) [Cortes and Vapnik, 1995] and the third one, referred to as SLAND, uses lazy association rules [Agrawal et al., 1993]. The three author grouping methods are those proposed by Han et al. [2005b], referred to as KWAY, by Huang et al. [2006], referred to as LASVM-DBSCAN and the state-of-the-art author grouping method known as HHC [Cota et al., 2010].

4.2.1 Experimental Setup

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

Each competing method was executed ten times and in each execution a different shuffling configuration was used[6]. 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.

Table 4.1. Results (with their standard deviations) obtained by the author grouping step for each ambiguous group in the (a) DBLP and (b) BDBComp collections, without using the popular last names.

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

(a) DBLP Collection

Ambiguous Group	ACP	AAP	K	pP	pR	pF1
A Oliveira	1.000± 0.000	0.600± 0.008	0.774± 0.005	1.000± 0.000	0.245± 0.019	0.394± 0.025
A Silva	1.000± 0.000	0.838± 0.022	0.915± 0.012	1.000± 0.000	0.511± 0.084	0.673± 0.074
F Silva	1.000± 0.000	0.914± 0.000	0.956± 0.000	1.000± 0.000	0.500± 0.000	0.667± 0.000
J Oliveira	1.000± 0.000	0.810± 0.049	0.900± 0.027	1.000± 0.000	0.646± 0.104	0.781± 0.078
J Silva	1.000± 0.000	0.782± 0.017	0.884± 0.009	1.000± 0.000	0.457± 0.047	0.626± 0.045
J Souza	1.000± 0.000	0.560± 0.010	0.749± 0.007	1.000± 0.000	0.273± 0.017	0.428± 0.021
L Silva	1.000± 0.000	0.818± 0.000	0.905± 0.000	1.000± 0.000	0.515± 0.000	0.680± 0.000
M Silva	1.000± 0.000	0.857± 0.000	0.926± 0.000	1.000± 0.000	0.400± 0.000	0.571± 0.000
R Santos	1.000± 0.000	0.950± 0.000	0.975± 0.000	1.000± 0.000	0.667± 0.000	0.800± 0.000
R Silva	0.963± 0.000	0.926± 0.000	0.944± 0.000	0.800± 0.000	0.667± 0.000	0.727± 0.000

(b) BDBComp collection

4.2.2 Evaluating the Author Grouping Step

We show in Tables 4.1 and 4.2 the results obtained by each strategy for the author grouping step, i.e., when the list of popular last names is not used (Table 4.1) and when it is explored to enforce additional constraints (Table 4.2) with the goal of increasing the purity of the clusters.

We notice that, by exploiting the list of popular last names, SAND produces purer clusters as hypothesized. When this list is not used, there are seven ambiguous

[6]We did this because the performance of the evaluated methods could, in thesis, be impacted by the order in which references are processed. As we shall see, this did not happen.

Table 4.2. Results (with their standard deviations) obtained by the author grouping step for each ambiguous group in the (a) DBLP and (b) BDBComp collections, using the popular last names.

Ambiguous Group	ACP	AAP	K	pP	pR	pF1
A Gupta	0.990 ± 0.002	0.429 ± 0.030	0.651 ± 0.023	0.994 ± 0.001	0.427 ± 0.051	0.596 ± 0.053
A Kumar	1.000 ± 0.000	0.241 ± 0.013	0.491 ± 0.013	1.000 ± 0.000	0.097 ± 0.007	0.176 ± 0.011
C Chen	0.950 ± 0.004	0.260 ± 0.004	0.497 ± 0.005	0.843 ± 0.031	0.087 ± 0.003	0.158 ± 0.005
D Johnson	1.000 ± 0.000	0.274 ± 0.033	0.523 ± 0.032	1.000 ± 0.000	0.253 ± 0.059	0.401 ± 0.078
J Martin	1.000 ± 0.000	0.508 ± 0.004	0.713 ± 0.003	1.000 ± 0.000	0.320 ± 0.002	0.485 ± 0.002
J Robinson	1.000 ± 0.000	0.347 ± 0.016	0.589 ± 0.014	1.000 ± 0.000	0.279 ± 0.020	0.435 ± 0.025
J Smith	0.987 ± 0.004	0.200 ± 0.030	0.443 ± 0.033	0.993 ± 0.005	0.186 ± 0.042	0.312 ± 0.059
K Tanaka	1.000 ± 0.000	0.378 ± 0.017	0.615 ± 0.014	1.000 ± 0.000	0.231 ± 0.013	0.374 ± 0.017
M Brown	1.000 ± 0.000	0.368 ± 0.000	0.607 ± 0.000	1.000 ± 0.000	0.301 ± 0.000	0.463 ± 0.000
M Jones	1.000 ± 0.000	0.266 ± 0.017	0.516 ± 0.017	1.000 ± 0.000	0.238 ± 0.023	0.383 ± 0.031
M Miller	0.993 ± 0.004	0.589 ± 0.015	0.765 ± 0.010	0.989 ± 0.008	0.575 ± 0.022	0.727 ± 0.019

(a) DBLP collection

Ambiguous Group	ACP	AAP	K	pP	pR	pF1
A Oliveira	1.000 ± 0.000	0.598 ± 0.006	0.773 ± 0.004	1.000 ± 0.000	0.241 ± 0.015	0.388 ± 0.019
A Silva	1.000 ± 0.000	0.835 ± 0.019	0.914 ± 0.011	1.000 ± 0.000	0.534 ± 0.083	0.692 ± 0.073
F Silva	1.000 ± 0.000	0.914 ± 0.000	0.956 ± 0.000	1.000 ± 0.000	0.500 ± 0.000	0.667 ± 0.000
J Oliveira	1.000 ± 0.000	0.838 ± 0.061	0.915 ± 0.034	1.000 ± 0.000	0.705 ± 0.130	0.821 ± 0.091
J Silva	1.000 ± 0.000	0.753 ± 0.017	0.868 ± 0.010	1.000 ± 0.000	0.439 ± 0.047	0.609 ± 0.046
J Souza	1.000 ± 0.000	0.509 ± 0.008	0.713 ± 0.005	1.000 ± 0.000	0.258 ± 0.013	0.409 ± 0.016
L Silva	1.000 ± 0.000	0.818 ± 0.000	0.905 ± 0.000	1.000 ± 0.000	0.515 ± 0.000	0.680 ± 0.000
M Silva	1.000 ± 0.000	0.857 ± 0.000	0.926 ± 0.000	1.000 ± 0.000	0.400 ± 0.000	0.571 ± 0.000
R Santos	1.000 ± 0.000	0.950 ± 0.000	0.975 ± 0.000	1.000 ± 0.000	0.667 ± 0.000	0.800 ± 0.000
R Silva	1.000 ± 0.000	0.916 ± 0.021	0.957 ± 0.011	1.000 ± 0.000	0.600 ± 0.141	0.740 ± 0.126

(b) BDBComp collection

groups in both collections with unpure clusters (i.e., ACP smaller than 1), while there are only four of these unpure ambiguous groups when SAND uses the list of popular last names.

From now on, all reported results will use the stronger strategy that exploits the list of popular last names in the author grouping step.

4.2.3 Evaluating the Clustering Selection Step

We evaluate several options for selecting the clusters whose references will compose the training data, including the use of two reference-based similarity metrics (i.e., cosine and euclidian distance) along with several clustering similarity strategies, detailed in Section 4.1.2, based on cluster centroids and single, complete and average linkages.

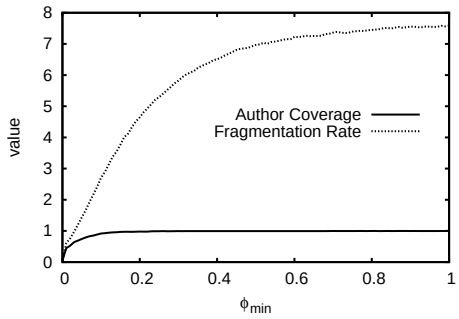
To evaluate the clustering selection step we use two metrics: author coverage and fragmentation rate. *Author coverage* measures the coverage of all real authors in the collection by the training data. This metric varies between [0,1] and achieves its peak when all authors in the collection have at least one representative cluster in the

training set. *Fragmentation rate* indicates the level of fragmentation of the references to a same author in the training set which grows from 0 (no clusters in the training data) to the total number of references in the collection (one reference per cluster) divided by the number of real authors. Ideally these two metrics should converge to 1, i.e., we should have only one cluster per author in the training set and all authors should be represented there.

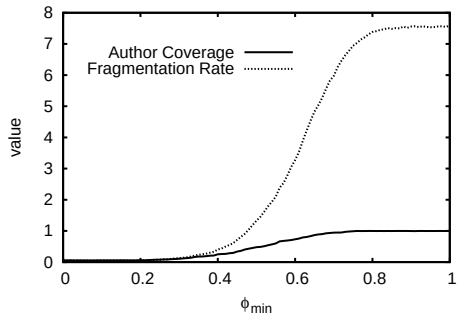
$$Author\ coverage = \frac{\#\ of\ different\ authors\ represented\ in\ the\ training\ data}{\#\ of\ real\ authors}$$

$$Fragmentation\ rate = \frac{\#\ of\ selected\ clusters}{\#\ of\ real\ authors}$$

DBLP Collection

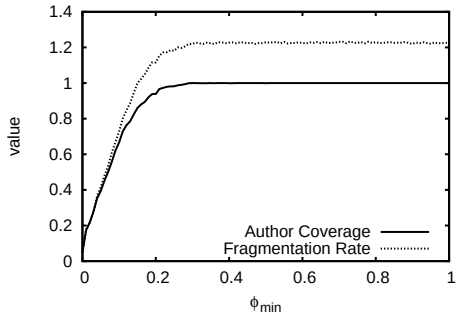


(a) Centroid – cosine

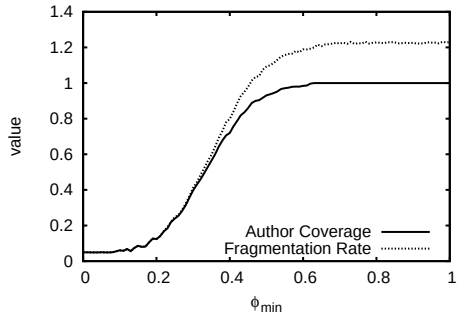


(b) Centroid – euclidian distance

BDBComp Collection



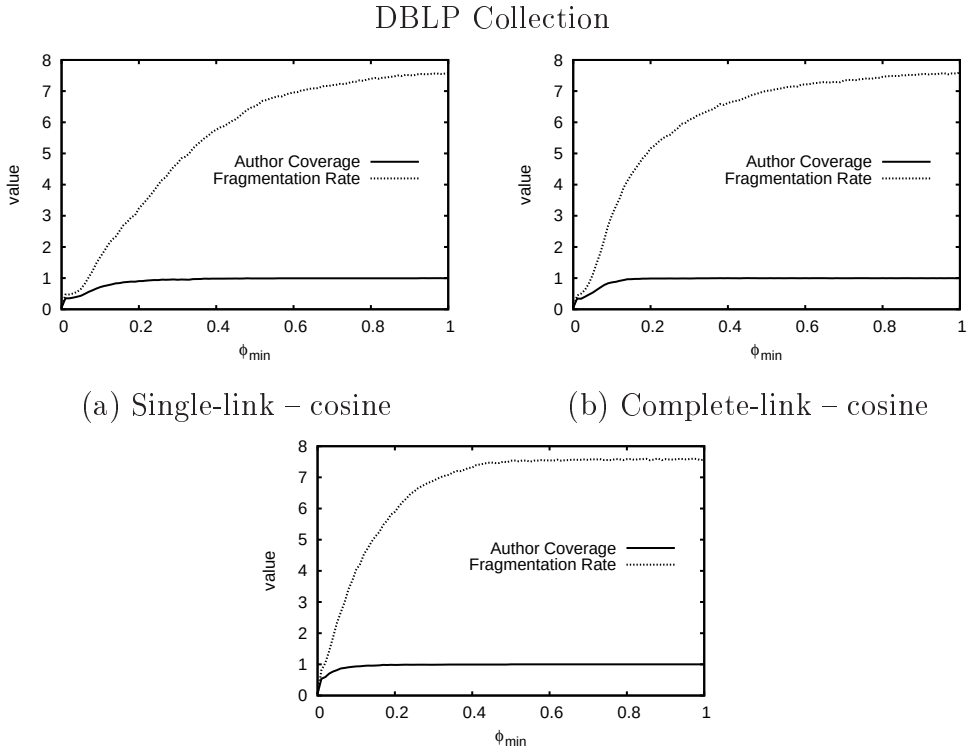
(c) Centroid – cosine



(d) Centroid – euclidean distance

Figure 4.2. Comparison between the cosine similarity function, (a) and (c), and euclidean distance, (b) and (d), for selecting the training data in DBLP and BDBComp.

Figures 4.2 (a–d) show the evolution of these two metrics in DBLP and BDBComp as the value of ϕ_{min} increases, using the cosine similarity function and the euclidean



(a) Single-link – cosine

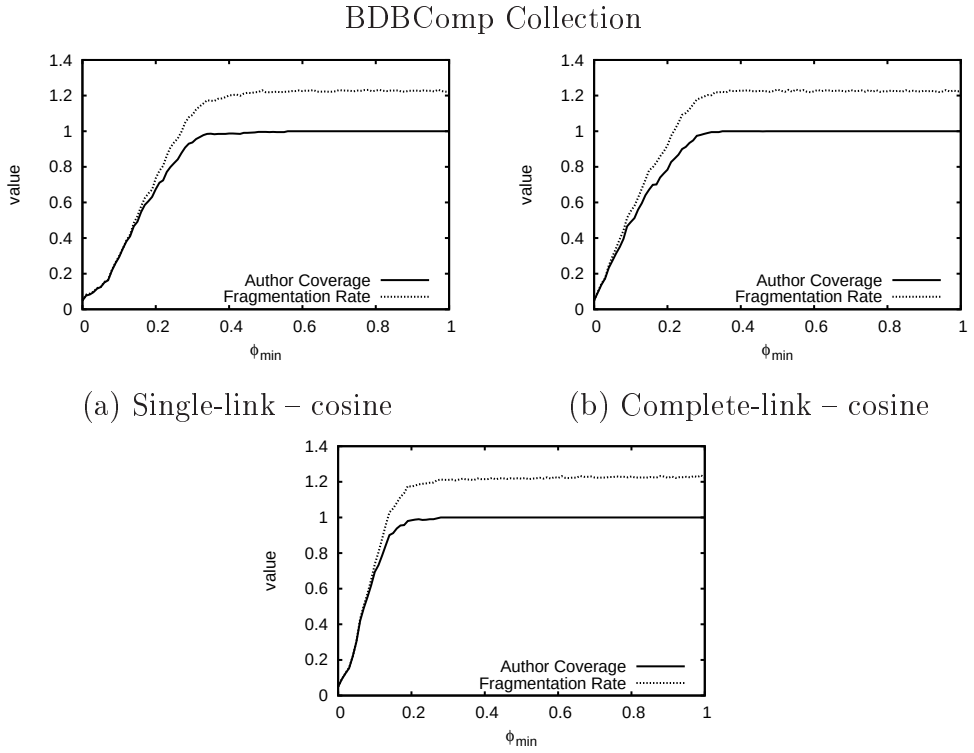
(b) Complete-link – cosine

(c) Average-link – cosine

Figure 4.3. Comparison between the author coverage and the fragmentation rate in DBLP using some strategies for selecting the training data. The selection of the training data uses (a) single-link, (b) complete-link and (c) average-link cluster similarities with cosine similarity function on the vectors.

distance as similarity functions applied to the centroids of the clusters. Looking at the figures, we can notice that, as expected, when we increase the value of the similarity threshold, ϕ_{min} (i.e., the minimum similarity value required for two clusters to be considered similar), we increase the fragmentation in the training data in both collections because more clusters are considered as being dissimilar. The increase in fragmentation in the training data affects the performance of our disambiguator since it will consider information of one real author as belonging to different authors labels in the training set.

As mentioned before, the ideal situation is when the number of authors in the training data rapidly approaches the number of real authors in the collections and this number is not much different from the number of selected clusters. For instance, when fragmentation is large we may have to select many clusters in order to have a good author coverage in the training data. Accordingly, we want to find out which combination of reference, author, and cluster similarities converges faster without selecting too many clusters.



(a) Single-link – cosine

(b) Complete-link – cosine

(c) Average-link – cosine

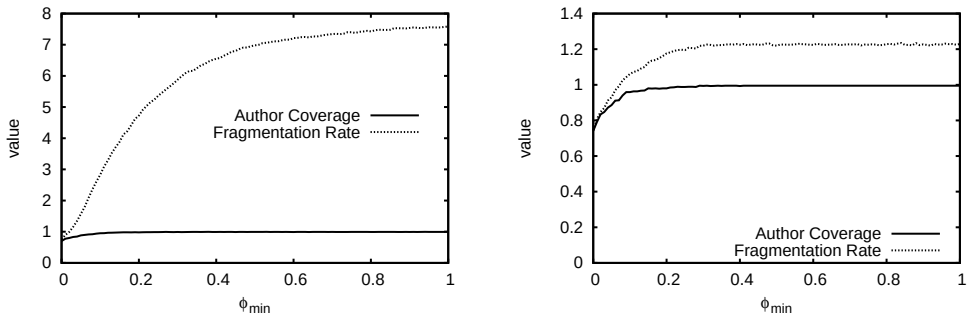
Figure 4.4. Comparison between the author coverage and the fragmentation rate in BDBComp using some strategies for selecting the training data. The selection of the training data uses (a) single-link, (b) complete-link and (c) average-link cluster similarities with cosine similarity function on the vectors.

Looking again at Figure 4.2, we can notice that by using the cosine similarity function the *author coverage* converges to 1 much faster than when we use the euclidian distance in both collections. Notice also that, at the first point when the author coverage achieves its maximum in DBLP, the fragmentation rate is considerably smaller when using cosine (around 3) than when we use the euclidian distance (around 7). A similar behavior is found with other combinations of clustering similarity techniques. Thus, given its evident superiority, in the remainder of the discussion we will always use cosine as similarity function.

We now turn our attention to the clustering based similarity techniques, using DBLP (see Figure 4.3). We recall that those techniques are single (see Figure 4.3 (a)), complete (see Figure 4.3 (b)) and average linkage (see Figure 4.3 (c)). We can see in the Figure 4.3 that the performance of these three cluster similarity techniques does not outperform the strategy that selects the training data using cosine similarity function applied to the centroids of the clusters (see Figure 4.2 (a)). For instance, single-link and complete-link converges to full coverage very slowly and, although average-link has

a faster convergence, its fragmentation grows much faster. Similar results are obtained for BDBComp, although the difference between average-link and the use of centroid is not so prominent. Given these results, from now on we will use only cluster centroids to measure cluster similarity.

Finally, in Figure 4.5 we show the performance of Strategy 3, i.e., when we combine a comparison on author names using fragment comparison with the cosine similarity function on the cluster centroids. Remind that, in this case, a cluster is selected to compose the training data when its author name is not similar to any author name of the selected clusters already in the training data or when its centroid is dissimilar to all previously selected clusters.



(a) DBLP (b) BDBComp

Figure 4.5. Strategy 3 performed in the (a) DBLP and (b) BDBComp collections.

We can notice in Figure 4.5 (a) that the author coverage and the fragmentation rate are already very close to 1 when $\phi_{min} = 0$, which is equivalent to use only the fragment comparison algorithm on the author names to select the clusters to compose the training data (or Strategy 2)[7]. Thus, selecting the clusters using only the comparison between the author names is a very good option, which, besides producing good results, does not incur in the costs of comparing all attributes of the references with a similarity metric. A similar situation occurs in BDBComp (see Figure 4.5 (b)) although its convergence to 1 is a bit slower[8]. Finally, notice in Figure 4.5 (b) that fragmentation does not increase as fast as in DBLP due to the fact that number of references per author in BDBComp is smaller than in DBLP.

In sum, although simple, a good option for selecting the clusters to compose the training data, at least in the collections we analyzed, is using only the fragment

[7]Strictly speaking, the best result is obtained with $\phi_{min} = 0.02$, but this is very close to using $\phi_{min} = 0$

[8]Notice also that results are already high in the beginning.

comparison algorithm for comparing author names, which produces fragmentation rate and author coverage close to 1.

4.2.4 Evaluating SAND

In this section, we discuss the final performance of SAND considering two situations. In the first one, referred to as SAND-1, we use the following configuration: (1) the author grouping step does not use popular last names, (2) the cluster selection step uses Strategy 1 with the cosine similarity function applied to the cluster centroids, and (3) the author assignment step predicts the author of each single reference. We use this configuration for comparative purposes as it corresponds to an early version of SAND [Ferreira et al., 2010].

In the second situation, refereed to as SAND-2, we exploit our best found configuration, i.e., (1) the author grouping step uses popular last names to increase purity, (2) the cluster selection step uses Strategy 3 with the fragment comparison algorithm applied to the author names and the cosine similarity function applied to the cluster centroids to select the clusters that will compose the training set, and (3) the author assignment step predicts the author of each cluster in the test set instead of each single reference (i.e., all references in a cluster are assigned to the same author).

Figure 4.6 shows the disambiguation performance of SAND considering both configurations for various values of ϕ_{min}. When using SAND-1, lower values of ϕ_{min} result in the selection of only few clusters in the training set, that is, important clusters may be not included in the training data. On the other hand, higher values of ϕ_{min} may result in the selection of several fragmented clusters. This happens because with higher vales of ϕ_{min} many clusters are consider as dissimilar. This can be easily seen by the sharp decrease in cohesion (or increase in fragmentation) values in Figure 4.6 for SAND-1 in both collections as we increase ϕ_{min}. Therefore this tradeoff needs to be carefully addressed in order to choose a suitable value for ϕ_{min} that maximizes performance in this configuration. This tradeoff is also seem in the values of the K metric. Particularly, in DBLP it increases as ϕ_{min} gets higher but after it peaks it starts droping by due to the sharp decrease in cohesion (i.e., increase in fragmentation). In BDBComp, there is a initial drop in K but then it starts to keep growing, and it tends to remain stable after ϕ_{min}=0.15.

With SAND-2, lower values (usually between 0 and 0.05) of ϕ_{min} already produce a good author coverage, i.e., the number of authors in the training set is close to the number of real authors in the collection as discussed in the previous Section, with low fragmentation. The decrease of performance of SAND-2 as we increase the value of

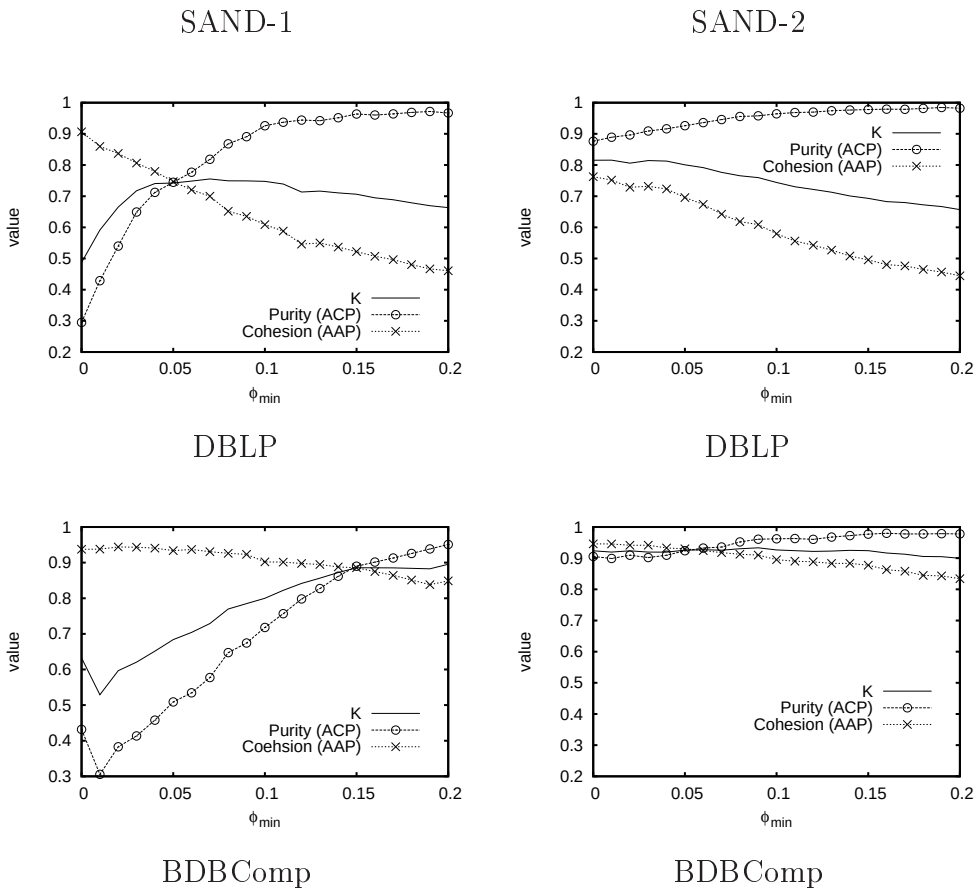


Figure 4.6. Sensitivity analysis for ϕ_{min}.

ϕ_{min}, in both collections, as observed before, is due to the increase of fragmentation in the training data. We also notice that the performance obtained by SAND-2 without using the ϕ_{min} (i.e., $\phi_{min} = 0$) is better that the best performance of SAND-1 in both collections. Thus, we set the ϕ_{min}=0 for SAND-2 in the next analyses, i.e., *we execute SAND without any parameter setup.*

Finally, Tables 4.3 and 4.4 show, respectively, the disambiguation performance for SAND-1 when it achieves its peak considering the K metric (i.e., $\phi_{min} = 0.07$ in DBLP and $\phi_{min} = 0.20$ in BDBComp) and for SAND-2 without using ϕ_{min} (i.e., $\phi_{max} = 0$) in each ambiguous group, averaged over the 10 executions (i.e., the shuffles). As we can see, the low standard deviations of these results mean that the shuffling configuration, that is, the order in which references are processed, does not affect much the results. We can also see that groups such as "C. Chen" and "R. Silva" are harder to disambiguate, mostly because the high ambiguity and large number of candidate authors in these groups. More importantly, we can see that SAND-2 outperforms SAND-1 in every single ambiguous group in both metrics in both collections. In average in the DBLP

Table 4.3. Results obtained by SAND-1.

Ambiguous Group	K	pF1
A. Gupta	0.768 ±0.029	0.721 ±0.049
A. Kumar	0.678 ±0.018	0.546 ±0.040
C. Chen	0.551 ±0.018	0.353 ±0.022
D. Johnson	0.679 ±0.014	0.667 ±0.014
J. Martin	0.835 ±0.015	0.747 ±0.022
J. Robinson	0.735 ±0.012	0.676 ±0.025
J. Smith	0.756 ±0.018	0.755 ±0.017
K. Tanaka	0.782 ±0.010	0.701 ±0.021
M. Brown	0.823 ±0.015	0.760 ±0.050
M. Jones	0.778 ±0.015	0.765 ±0.021
M. Miller	0.898 ±0.013	0.919 ±0.013
Average	0.753±0.005	0.692±0.009

(a) DBLP

Ambiguous Group	K	pF1
A. Oliveira	0.847± 0.030	0.710± 0.105
A. Silva	0.947± 0.015	0.835± 0.084
F. Silva	0.954± 0.000	0.714± 0.000
J. Oliveira	0.917± 0.026	0.869± 0.052
J. Silva	0.911± 0.030	0.721± 0.068
J. Souza	0.751± 0.006	0.435± 0.019
L. Silva	0.844± 0.034	0.597± 0.093
M. Silva	0.926± 0.000	0.571± 0.000
R. Santos	0.975± 0.000	0.800± 0.000
R. Silva	0.895± 0.018	0.551± 0.056
Average	0.897±0.003	0.680±0.013

(b) BDBComp

collection, SAND-2 outperforms SAND-1 in more than 8% under the K metric and 15% under the pF1 metric. In BDBComp collection, SAND-2 outperforms SAND-1 in more than 3% under K metric and 10% under the pF1 metric.

Therefore, given the much improved performance of SAND-2 and the fact that it does not need any parameter setup, from now on we will consider this configuration in all further analyses.

4.2.5 Comparison with the Author Grouping Baselines

Table 4.5 shows the comparison of SAND with its best configuration against three representative author grouping methods: KWAY, LASVM-DBSCAN and HHC. For KWAY, we used the implementation of the K-way spectral clustering provided by the University of Washington Spectral Clustering Working Group[9]. For LASVM-DBSCAN, we

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

Table 4.4. Results obtained by SAND-2.

Ambiguous Group	K	pF1
A. Gupta	0.865±0.025	0.883±0.036
A. Kumar	0.784±0.071	0.719±0.120
C. Chen	0.649±0.022	0.514±0.040
D. Johnson	0.741±0.062	0.733±0.138
J. Martin	0.863±0.037	0.820±0.067
J. Robinson	0.822±0.018	0.820±0.020
J. Smith	0.762±0.033	0.739±0.072
K. Tanaka	0.889±0.016	0.912±0.026
M. Brown	0.900±0.008	0.920±0.004
M. Jones	0.780±0.033	0.769±0.047
M. Miller	0.912±0.016	0.931±0.020
Average	0.815±0.010	0.796±0.020

(a) DBLP

Ambiguous Group	K	pF1
A Oliveira	0.930±0.041	0.903±0.098
A Silva	0.982±0.012	0.971±0.032
F Silva	0.954±0.000	0.714±0.000
J Oliveira	0.825±0.036	0.676±0.087
J Silva	0.951±0.017	0.929±0.015
J Souza	0.938±0.015	0.904±0.032
L Silva	0.901±0.024	0.737±0.080
M Silva	0.948±0.010	0.735±0.047
R Santos	0.911±0.011	0.480±0.042
R Silva	0.896±0.015	0.471±0.058
Average	0.924±0.004	0.752±0.015

(b) BDBComp

used the LaSVM package [Bordes et al., 2005] and the DBSCAN version available from Weka[10]. For HHC, we used our own implementation of the method.

Results show that, in the DBLP collection, SAND outperforms all author grouping baselines. Gains range from 45% (against LASVM-DBSCAN) to 5.4% in terms of the K metric, and 96% (against KWAY) to 6% in terms of pairwise F1. In the BDBComp collection, SAND outperforms KWAY and LASVM-DBSCAN by more than 14% and 72% under the K and pF1 metrics, respectively, and is statistically tied with HHC.

Particularly, the poor performance of LASVM-DBSCAN is mainly due to the small number of attributes used when compared with the original proposed method described in [Huang et al., 2006]. In that work, several other attributes such as affiliation and e-mail were used. In the scenario of author name disambiguation in which only the few most common attributes are available (the scenario we focus here as it is

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

Table 4.5. Results obtained by SAND, HHC, KWAY and LASVM-DBSCAN methods. Best results are highlighted in bold.

Method	K	pF1
SAND	**0.815**	**0.796**
HHC	0.773	0.751
KWAY	0.560	0.402
LASVM-DBSCAN	0.551	0.406

(a) DBLP

Method	K	pF1
SAND	**0.924**	**0.752**
HHC	**0.913**	**0.756**
KWAY	0.805	0.436
LASVM-DBSCAN	0.757	0.211

(b) BDBComp

the most common one), the similarity functions learned by the LASVM-DBSCAN are not suitably generalizable.

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 LASVM-DBSCAN, though possibly incurring in more false positive and negative errors (i.e., wrong assignments). HHC, the strongest author grouping baseline, uses some heuristics also used by us (e.g., clustering by coauthor), but those were improved, for example, to guarantee even purer clusters for training. HHC also does not include a supervised second step.

SAND, on the other hand, was able to produce better results, being able to predict the correct author of a given record using disambiguation functions learned from examples automatically selected.

Next we compare SAND with some supervised author assignment methods that can also take advantage of learning from training examples, although in their case, the examples were manually labeled.

4.2.6 Comparison with the Supervised Author Assignment Methods

In this Section we compare SAND with three representative author assignment methods, namely SVM, NB, and SLAND. RBF kernels were used for SVM and we employed the LibSVM tool [Chang and Lin, 2001] for finding their optimum parameters for each

training data on each ambiguous group. We estimate the parameter of the NB method as in [Han et al., 2004]. For SLAND, the state-of-the-art supervised author assignment method, the best parameters were discovered using cross-validation in the training set.

For the tests in this section, each ambiguous group was randomly split into training (50%) and test (50%) sets. This split ensures a fair comparison among these methods. It should be noticed that SAND is executed only with the test sets. All results shown next correspond to the performance in the test sets and are the average of 10 runs. The results are compared using statistical significance tests (paired t-test) with 99% confidence interval.

It is very important to stress that while the baselines used the whole manually labeled training sets to learn the disambiguation functions and apply them to the test sets, we did not use this information and automatically generated the training data by applying SAND *directly to the test data* in each round, making no use of manually assigned labels.

Table 4.6. Results (with their standard deviations) of SAND, SLAND, SVM and NB in the DBLP and BDBComp collections. Best results, including statistical ties, are highlighted in bold.

Method	K	pF1
SAND	0.775±0.010	0.720±0.018
SLAND	**0.877**±0.007	**0.867**±0.008
SVM	0.799±0.008	0.721±0.010
NB	0.736±0.009	0.647±0.012

(a) DBLP

Method	K	pF1
SAND	**0.940**±0.014	**0.462**±0.040
SLAND	0.900±0.016	**0.456**±0.028
SVM	0.481±0.024	0.160±0.032
NB	0.420±0.009	0.160±0.019

(b) BDBComp

As it can be seen, in DBLP (see Table 4.6(a)), SLAND achieves statistically superior results, but SAND results are only 11.6% and 17% lower than the ones obtained by SLAND, under K and pF1, respectively, without any manually labeled training data. Furthermore, SAND largely outperforms NB and has basically the same performance as SVM (slightly inferior according to the K metrics (approximately 3%) and statistically tied under the pF1 metric).

In BDBComp, results are quite surprising. SAND outperforms SVM and NB under both metrics by more than 95% and is even superior to SLAND under the K metric. The BDBComp collection has several authors with only one or two publication. Selecting 50% of the data to compose the training data does not ensure that all authors have at least one example in the training data. On the other hand, the coverage of the training data automatically constructed by SAND applied directly to the test set may be more representative than the ones used by baselines. Furthermore, in this collection, the heuristics used in the author grouping step based on co-author names are very effective and enough to solve a large number of cases, leaving for the other SAND steps only a refined adjustment. Notice that these adjustments can further explore interesting properties, such as the detection of new authors and self-training. This shows another interesting capabilities of our solution that may be explored in collections with similar characteristics.

We also run SLAND with reduced training data to check its performance when a smaller number of references are labeled in the DBLP collection. We randomly pick up 10%, 20%, 30%, 40% and 50% of the training data and run SLAND with these new training examples. For each original training data, we picked up the examples 10 times, performed SLAND and averaged the results. With 10%, 20%, 30%, 40% and 50% of the training data, the K values were 0.615, 0.700, 0.758, 0.786 and 0.824, respectively. Notice that, labeling until 30% of the training data, the performance of SLAND is worse than SAND without labeling any example.

4.2.7 Comparison with Other Supervised Methods for the Author Assignment Step

To check whether our choice of classifier was the best for the supervised step, we run experiments in which the associative classifier was replaced by the corresponding ones used in baselines, i.e., we used other learning algorithms in the author assignment step. Specifically, we evaluated the application of SVM and Naïve Bayes in the author assignment step. Here, we called S-SVM and S-NB the application of SVM and Naïve Bayes, respectively, in the author assignment step. Table 4.7 shows the results. We notice that the original SAND outperforms all competitors in DBLP by more than 27% in terms of the K metrics and more than 70% under pF1, and is statistically tied with S-SVM in BDBComp under K metric and far superior under pF1 (gains of 82%), being also much better than S-NB is this collection under both metrics. The superiority of original SAND configuration, mainly in DBLP, is probably due to its capability of adding new examples based on reliable predictions, and also of identifying new authors

not present in the provided training data.

Table 4.7. Results obtained by the author grouping and cluster selection steps coupled with SVMs (S-SVM) and Naïve Bayes (S-NB) techniques in the second step (i.e., the author assignment step). Best results are highlighted in bold.

Method	K	pF1
SAND	**0.815**±0.010	**0.796**±0.020
S-SVM	0.666±0.009	0.489±0.018
S-NB	0.640±0.014	0.466±0.026

(a) DBLP

Method	K	pF1
SAND	**0.924**±0.004	**0.752**±0.015
S-SVM	**0.917**±0.006	0.412±0.020
S-NB	0.883±0.013	0.286±0.037

(b) BDBComp

Regarding efficiency issues of proposed method, we measured the time spent to infer the author of each reference in the test set. On average, the time to assign each reference to its author was around 0.2 second. We perform our evaluation in a Intel Xeon E5620 with 2.40GHz and 8 gigabytes of RAM. This means that we could disambiguate approximately half million records in one day.

4.2.8 Discussion

In this Section, we further discuss two additional but important issues: the impact in our method of the absence of long format author names in the collections and the relative performance of SAND (and its competitors) over time.

4.2.8.1 How effective is SAND when all author names are in short format?

We evaluate the performance of SAND in its best configuration when all author names are short (i.e., all author names have only the initial of the first name and the full last name). We do this by substituting all the author names in long format by their short version in our collections. This corresponds to an extreme ambiguous situation which may push our method to its limits.

Results are shown in the Figure 4.7. Remember that around 53% and only 3% of the author names in the original DBLP and BDBComp collections, respectively, are in short format. We can see in the Figure that, when ϕ_{min} is equal to 0, as in the previous

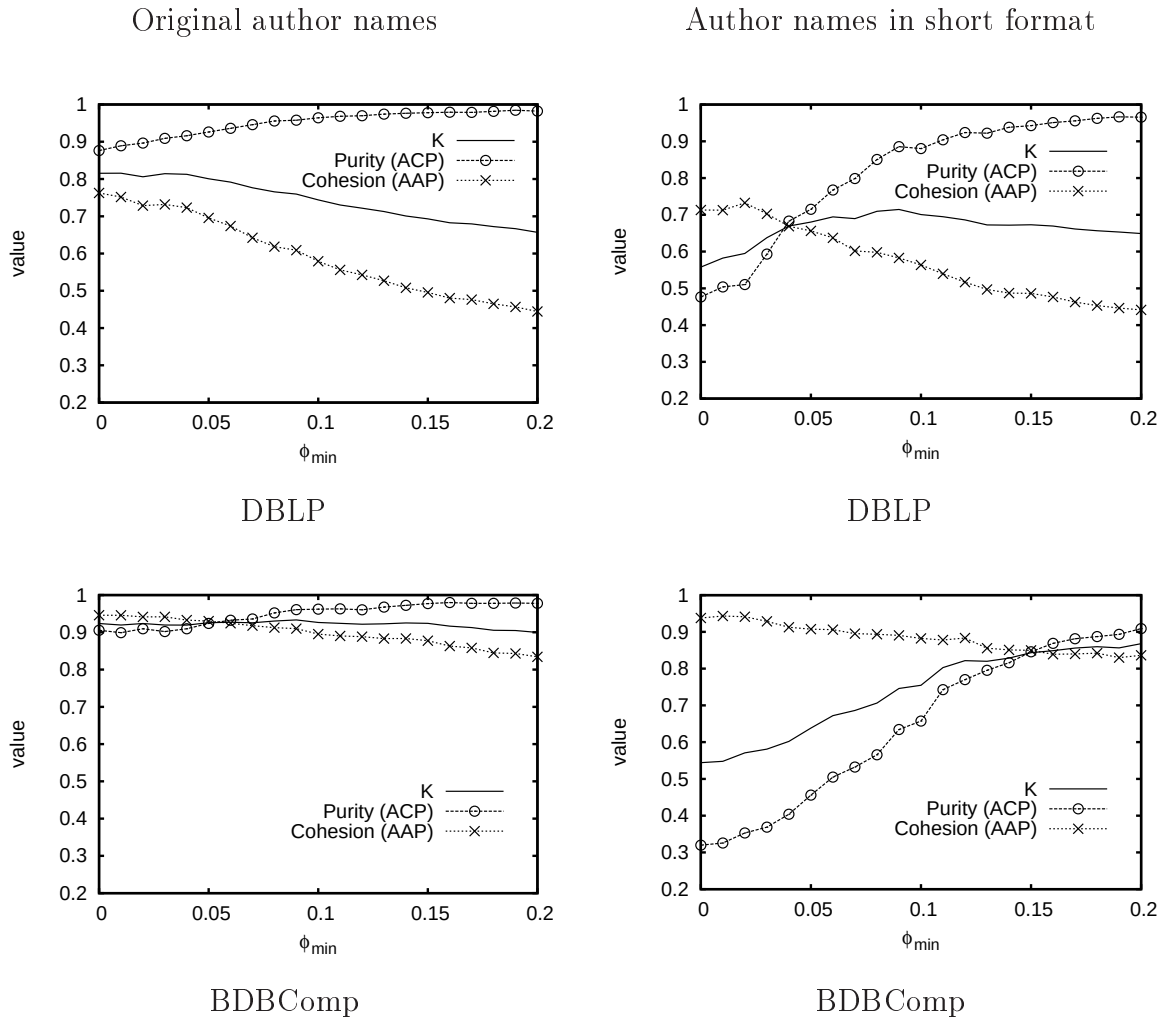


Figure 4.7. Sensitivity analysis for ϕ_{min}. The comparison of SAND's performance using the name of the authors as provided in the collections with the author names in short format (i.e., the author names are represented by only the initial of first name and the full last name).

configuration, we obtain worse results than in the original collections, as expected. In DBLP the performance loss is around 31% under the K metric, while for BDBComp the losses are around 41% . However, as it can been seen in the figure, if we use the best suitable configuration for this situation in DBLP (ϕ_{min}=0.09 and K=0.715) and BDBComp (ϕ_{min}=0.16 and K=0.842), we can reduce the losses to 12% and 8% in both collections, respectively. In summary, the format of the author names, as expected, affects the results of SAND. The larger the number of author names in short format, the higher the ambiguity, and therefore the more difficult the disambiguation process is. However, we can deal with these difficulties, at least in part, by properly choosing the ϕ_{min} parameter.

4.2.8.2 How effective is SAND and the baselines over time?

To evaluate the behavior of SAND and the baselines over time, we use collections generated by SyGAR. We used the same scenarios and performed similar experiments as in Section 5.3 to compare the performance of our method against the best author grouping and author assignment baselines.

Our evaluation was carried out by computing the value of the K metric at each state of the DL. The results reported in the following sections are averages of 5 runs. Figure 4.8 shows, for each disambiguation method, the average K value computed over all 11 ambiguous groups in each state of the digital library over the ten-year load period in Scenario 1 described in the previous chapter, i.e., an evolving DL with static author population and static author profiles. The corresponding 95% confidence intervals are also shown.

Notice that state 0 corresponds to the first new load into the DL. The supervised author assignment methods (SLAND and SVM) are trained with a whole synthetic DL that was generated before the first load. SLAND and the unsupervised author grouping methods (HHC and KWAY) act only over the new states of the DL without any training. Notice also that we do not retrain SLAND and SVM after the new loads.

Looking at Figure 4.8 we can see that SLAND, which used the whole disambiguated DL for learning previously to the first load, is the best performer, as expected, followed by SAND and HHC which are basically tied throughout the whole loading period in this scenario. Notice that these three methods keep their effectiveness quite stable over time. SVM, on the other hand, is statistically tied with SAND and HHC in the first two loads, but starts losing performance after the second load and keeps degrading after each new one. After 10 loads, the performance of SVM degrades by 15%. KWAY, on its turn, starts with a poor performance in the first load but experiences an increasing improvement as new citations are added, finishing, after 10 loads, with an effectiveness basically identical to SVM's but still far from the best three methods. This improvement occurs because there is incrementally more information about each author, helping KWAY to better characterize them.

We now analyze the impact on each method of introducing new authors to the current author population. Figure 4.9 shows average K values and corresponding 95% confidence intervals for each method for $\%_{NewAuthors}$ equals to 5% and 10%.

As we can see, all methods, except KWAY, follow a similar trend, i.e., all suffer some considerable performance degradation, while in this scenario KWAY actually improves in performance as new loads of citations are added to the DL, after having started with a very low performance. However, the improvement in performance pre-

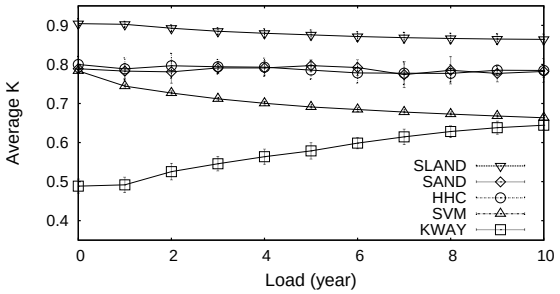
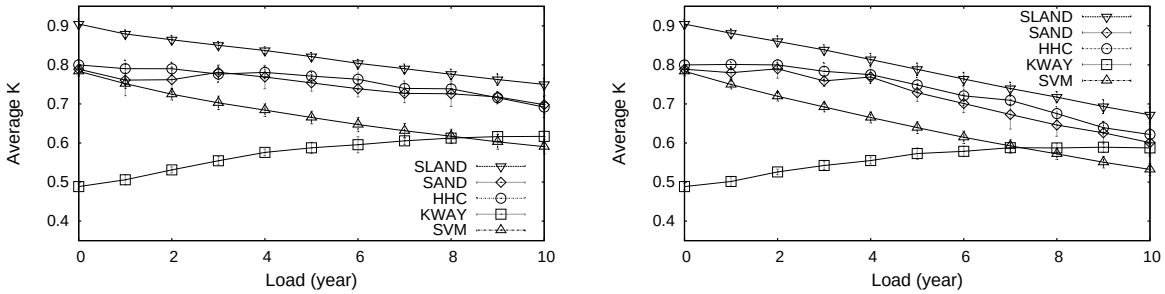


Figure 4.8. Scenario 1: Evolving DL with static author population and publication profiles.



(a) $\%_{NewAuthors}$=5% (b) $\%_{NewAuthors}$=10%

Figure 4.9. Scenario 2: Evolving DL and addition of new authors ($\%_{InheritedTopics}$=80%).

viously experienced by KWAY in the absence of new authors becomes less significant when $\%_{NewAuthors}$ increases. New ambiguous authors imply in higher ambiguity and, thus, a higher inherent difficulty in distinguishing them. We can also note that the performance of all other methods degrades much faster for $\%_{NewAuthors}$=10% than for $\%_{NewAuthors}$=5%, SVM being the most affected method with the introduction of new authors (much faster decay in both scenarios).

Comparing the methods, SLAND continues being the best performer in both scenarios when new authors are inserted, while SAND and HHC are basically tied, as in the previous scenario, throughout the whole period. After the last load, the differences in average performance between SLAND over SAND and HHC are of 7% and 8% for $\%_{NewAuthors}$=5% and $\%_{NewAuthors}$=10%, respectively. SLAND with $\%_{NewAuthors}$=5% and $\%_{NewAuthors}$=10% loses about 13% and 22%, respectively, in performance compared with the scenario with static author population. Comparing SAND and HHC with KWAY, the differences in performance are 12% for $\%_{NewAuthors}$=5% and a statistical tie for $\%_{NewAuthors}$=10%, while the difference is around 11% with static author population. SVM, on the other hand, gets statistically tied with KWAY after 10 loads in the

scenario with $\%_{NewAuthors}$=5% and is the worst performer of all for $\%_{NewAuthors}$=10%, significantly worse than all other methods.

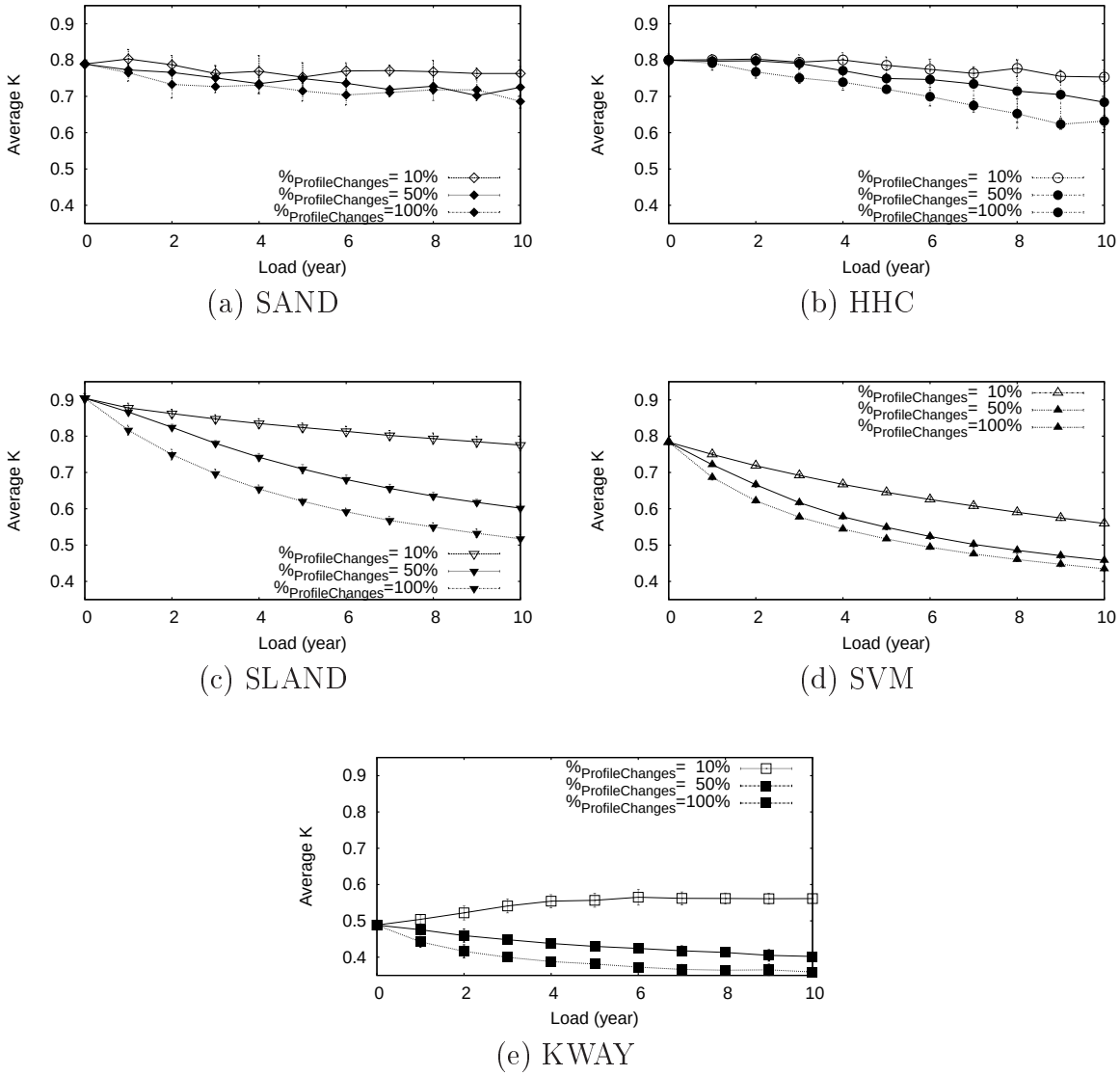


Figure 4.10. Scenario 3: Dynamic author profiles ($\delta = 5$ and $\%_{ProfileChanges}$=10%, 50% and 100%).

Finally, Figures 4.10(a-c) show average K values and corresponding 95% confidence intervals when a fraction $\%_{ProfileChanges}$ equal to 10%, 50% and 100% of the authors experience changes in their profiles at each new load. In this scenario, we can see a clear advantage of SAND over all methods, even over HHC, with which SAND was tied in the two previous scenarios. SAND is the most resilient method, independently of the level of profile change. After 10 loads, for instance, SAND's performance,

considering the averaged K values is 8% superior to HHC for $\%_{ProfileChanges}$ =100%. In fact, SAND is even superior to SLAND around 20% and 32% for $\%_{ProfileChanges}$ = 50% and 100%, respectively. We can also see that SVM performs poorly in this scenario and that KWAY, differently from previous scenarios, basically loses its capability of gaining in performance in face of profile changes.

Overall, among the three methods, considering all scenarios and all situations, SAND figured among the best performers in all analysed situations, being more affected by the introduction of new authors into the DL, an issue that can possibly be improved in future work.

Chapter 5

SyGAR: Synthetic Generator of Authorship Records

A solid analysis of existing methods should consider various scenarios that occur in real digital libraries. In addition to dynamic patterns, the analysis should also address the robustness of existing methods under data errors, such as typographical,optical character recognition (OCR) and speech recognition errors. However, the construction of a real, previously disambiguated, temporal collection capturing different relevant dynamic scenarios and including various data errors is quite costly. An alternative is to build realistic *synthetic collections* that capture all scenarios of interest, under controlled conditions, while still inheriting the properties of real collections that are more relevant from the standpoint of existing name disambiguation methods. In particular, a generator of realistic synthetic collections, designed for the specific problem of name ambiguity, should be able to:

- Generate data whose disambiguation is non-trivial, following patterns similar to those found in real collections;
- Generate successive loads of data, at a certain frequency (e.g., one per year or month), containing new publications of the same set of authors, to assess the impact of the introduction of new publications into previously disambiguated digital libraries on the disambiguation methods;
- Generate data for new authors that were not originally included in the collection, simulating the situation in which the disambiguation method must identify the appearance of publications of authors not yet present in the digital library;
- Generate data reflecting changes in the authors' publication profiles (e.g., changes

in the topics in which the authors publish), simulating changes of research interests over time;

- Introduce controlled errors on generated data, simulating errors caused by typos, misspelling, or OCR.

In order to address these requirements, in this chapter, we introduce and evaluate SyGAR, a new Synthetic Generator of Authorship Records, which addresses all the elicited requirements. SyGAR is capable of generating synthetic citation records given as input a list of disambiguated records of citations extracted from a real digital library (input collection). The synthetically generated records follow the publication profiles of existing authors, extracted from the input collection. An author's profile is generated based on a set of distributions, namely, distribution of the number of coauthors per record, distribution of coauthor popularity, distribution of number of terms in a work title as well as distribution of topic (subject or interest) popularity of the given author. Each topic is associated with term and venue popularity distributions. SyGAR can be parameterized to generate records for new authors (not present in the input collection), for authors with dynamic profiles, as well as records containing typographical errors. For the best of our knowledge, SyGAR is the first generator of its kind, enabling and facilitating the investigation of several aspects of existing name disambiguation methods.

A variety of synthetic data generators is available in the literature, most of them designed for a specific purpose. DSGen [Christen, 2005], for instance, is a tool to generate synthetic data representing personal information, such as first name, surname, address, dates, telephone and identifier numbers, which was developed as part of the Febrl deduplication system [Christen, 2008]. With that specific goal, DSGen generates synthetic data and duplicates them, inserting errors representing typographical errors. A more recent version of DSGen, introducing attribute dependencies as well as family and household data, is presented in [Christen and Pudjijono, 2009].

In contrast, there are also a few general-purpose tools, such as DGL [Bruno and Chaudhuri, 2005] and PSDG [Hoag and Thompson, 2007], which generate data based on specific languages used to describe several aspects of the data to be synthesized (e.g., distributions). These tools allow users to specify dependencies between attributes. However, neither of them can be parameterized with data from a given knowledge base, such as an existing real collection or a coauthorship graph. Such a feature is attractive as it can be exploited by the tool to infer attribute distributions from real-world data.

We are aware of only two synthetic data generators in the realm of digital libraries. The first one, SearchGen [Li et al., 2007] generates synthetic workloads of scientific literature digital libraries and search engines. SearchGen was designed based on a characterization of the workload of CiteSeer[1], extracted from usage logs. Li et al. validated the proposed tool by comparing the workload generated by SearchGen against logged workloads. SearchGen is fundamentally different from SyGAR, as our tool does not target the generation of workloads but rather of ambiguous citation records.

A tool that is more closely related to ours is the two-stage data generator proposed in [Bhattacharya and Getoor, 2007]. The tool was designed to generate synthetic citations, specified by a list of authors. It works as follows. In the first stage, it builds a collaboration graph containing entities (i.e., authors) and relationships among them (i.e., coauthorships). In the second stage, it generates a collection of citations, each of which synthesized by first randomly selecting one entity and then randomly selecting its neighbors in the collaboration graph. SyGAR significantly differs from this tool. First, it generates values to other attributes, such as work and publication venue titles, in addition to author and coauthor names. Second, it is capable of generating a dynamically evolving collection, in which new authors, changes in author's publication profiles and typographical errors may be introduced, at various rates. As such, our generator can be used to generate and simulate several controlled, yet realistic, long term scenarios, enabling an assessment of how distinct methods would behave under various conditions.

A preliminary version of SyGAR was discussed in [Ferreira et al., 2009]. In that prior version, SyGAR modeled an author's publication profile based on the distributions of the number of coauthors, coauthor popularity, number of terms in a work title, term popularity and venue popularity. By associating term and venue popularity distributions directly with the authors, our preliminary approach restricts the generation of citations containing only terms and venues that have been previously used by the authors. In its current version, SyGAR does not include term and venue popularity distributions as part of an author's profile. Instead, the profile of an author contains a distribution of *topics* (or research interests), and each topic has term and venue popularity distributions associated with it. This allows the generation of citations with work titles containing terms that have never been used by the authors or with a venue in which the authors have never published before. Moreover, the present tool allows one to generate data reflecting changes in the authors' publication profiles, simulating changes of research interests over time, and to introduce controlled errors on generated

[1] *http://citeseer.ist.psu.edu*

data, simulating errors caused by typos, misspelling, or OCR. Thus, the present tool is much more sophisticated and provides much more flexibility and richness to the process of generating synthetic citation records than its prior version.

We validate SyGAR by comparing the results produced by three representative disambiguation methods on a standard real collection of (previously disambiguated) records and on synthetic collections produced using SyGAR parameterized with author profiles extracted from the real collection. The methods considered are: the supervised support vector machine-based method (SVM) proposed by Han et al. [2004], the hierarchical heuristic-based method (HHC) proposed by Cota et al. [2010] and the unsupervised k-way spectral clustering-based method (KWAY) proposed by Han et al. [2005b]. Our experiments show, for all three methods, a very good agreement in the performance obtained for real and synthetically generated collections, with relative performance differences under 10% in most cases.

To demonstrate the applicability of our generator, we evaluate the three aforementioned methods in three selected relevant real-world scenarios. In particular, we simulate a digital library evolving over a period of several years, during which (1) new publications of the same set of authors are introduced, (2) new authors with ambiguous names are introduced, at various rates, and (3) a fraction of the authors change their publication profiles. Our results indicate that the performance of SVM tends to degrade with time, particularly as new authors are introduced in the collection. In contrast, the performance of the unsupervised KWAY method, which uses privileged information regarding the number of authors in the digital library, tends to increase with time, except when there are changes in the authors' profiles. Overall, among the three methods, HHC has the best performance, due to its heuristic that was specially designed to address the name disambiguation task. In terms of their drawbacks, HHC suffers more with the addition of records of new authors, whereas SVM and KWAY are very sensitive to changes in the authors' profiles.

5.1 SyGAR Design

SyGAR is a tool for generating synthetic collections of citation records. Its design was driven by our goal of evaluating name disambiguation methods in more realistic, yet controlled, scenarios, with evolving digital libraries. It was thus designed to capture the aspects of real collections that are key to disambiguation methods and, therefore, to generate synthetic collections to evaluate them. These synthetic collections may be larger and span longer periods of time besides being representative of the real data

with respect to *author publication profiles* (defined below).

SyGAR takes as input a real collection of previously disambiguated citation records, referred to as the *input collection*. Each such record is composed of the three attributes commonly exploited by disambiguation methods [Cota et al., 2010; Ferreira et al., 2010; Han et al., 2004, 2005a,b; Lee et al., 2005; Pereira et al., 2009], namely, a list of author names and a list of unique terms present in the work title and the publication venue title. Authors with the same ambiguous name, and their corresponding records, are organized into *ambiguous groups* (e.g., all authors with name "C. Chen"). SyGAR also takes as input several other parameters, defined in Table 5.1 and described in the following sections, which are used in the data generation process.

Table 5.1. SyGAR input parameters.

Parameter	Description
N_{loads}	number of loads to be synthesized
N_R	total number of records to be generated per load
N_{Topics}	number of topics
α_{Topic}	threshold used to estimate distribution of topic popularity per citation (LDA model)
α_{Term}	threshold used to estimate distribution of term popularity per topic (LDA model)
β_{Topic}	minimum weight of topics that are associated with an author
$\alpha_{NewCoauthor}$	probability of selecting a *new* coauthor
$\alpha_{NewVenue}$	probability of selecting a *new* venue
$\%_{NewAuthors}$	percentage of new authors to be generated in each load
$\%_{InheritedTopics}$	percentage of topics to be inherited from a new author's main coauthor
$\%_{ProfileChanges}$	percentage of authors that will have changes in their profiles in each load
δ	shift parameter used to simulate changes in an author's profile
P^{FName}	probability distribution of altering (removing, keeping or retaining) only the initial of the author's first name
P^{MName}	probability distribution of altering (removing, keeping or retaining) only the initial of the author's middle name
$P^{LName}_{\#Mods}$	probability distribution of the number of modifications in the author's last name
P^{LName}_{Mod}	probabilities of inserting, deleting or changing one character or swapping two characters of the author's last name
$P^{Title}_{\#Mod}$	probability distribution of the number of modifications in the work title
P^{Title}_{Mod}	probabilities of inserting, deleting or changing one character or swapping two characters of the title
$P^{Venue}_{\#Mods}$	probability distribution of the number of modifications in the venue
P^{Venue}_{Mod}	probabilities of inserting, deleting or changing one character or swapping two characters of the venue

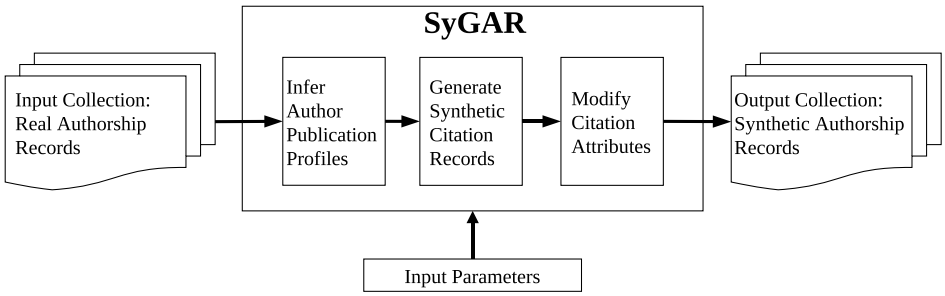


Figure 5.1. SyGAR main components – SyGAR receives as input a disambiguated collection of citation records and builds publication profiles for all authors in the input collection. Then, the publication profiles are used to generate synthetic records. As a final step, SyGAR may introduce typographical errors in the output collection and change the citation attributes.

As output, SyGAR produces a representative list of synthetically generated cita-

tion records, referred to as the corresponding *output collection*. Each generated record consists of the three aforementioned attributes. In particular, the (synthetic) work title is represented by a set of unique terms as opposed to a complete semantically-sound sentence, as most disambiguation methods typically exploit the former.

The overall generation process consists of three main steps, as shown in Figure 5.1. SyGAR first summarizes the input collection into a set of attribute distributions that characterize the publication profiles of individual authors in the input collection. SyGAR builds publication profiles for all authors in the input collection, including those with ambiguous and non-ambiguous names. Next, the attribute distributions are used to generate synthetic records. Unless otherwise noted, only profiles of authors *with ambiguous names* are used to generate synthetic data[2]. As a final step, SyGAR changes the citation attributes, particularly the coauthor names, so as to adhere to a pre-defined format (e.g., keep only the initial of the first name). In this step, it may also introduce typographical errors in the *output collection*. A detailed description of each step is presented in the following subsections.

5.1.1 Inferring Publication Profiles from the Input Collection

Each author's publication profile is characterized by her citation records. That is, the profile of author a is extracted from the input collection by summarizing her list of citation records into four probability distributions, namely:

1. a's distribution of number of coauthors per record - $P^a_{nCoauthors}$;
2. a's coauthor popularity distribution - $P^a_{Coauthor}$;
3. a's distribution of number of terms in a work title - P^a_{nTerms};
4. a's topic popularity distribution - P^a_{Topic}.

Each topic t is further characterized by two probability distributions:

1. t's term popularity distribution - P^t_{Term};
2. t's venue popularity distribution - P^t_{Venue}.

Finally, we also build a collection profile with:

1. probability distribution of the number of records per author with ambiguous names - $P^c_{nRecordsAuthors}$;

[2]Profiles of authors with non-ambiguous names are used in the generation of profiles of new authors (Section 5.1.3), which relies on the profiles of all authors in the input collection.

2. probability distribution of the number of records per author - $P^c_{nRecordsAllAuthors}$;

3. probability distribution of the number of records per ambiguous group - $P^c_{nRecordsGroup}$.

$P^a_{nCoauthors}$, $P^a_{Coauthor}$, P^a_{nTerms}, $P^c_{nRecordsAuthors}$ and $P^c_{nRecordsAllAuthors}$ can be directly extracted from the input collection by aggregating the citation records of each author a. We assume a's attribute distributions are statistically independent. In particular, we assume that, for any given citation, a's coauthors are independently chosen. Despite somewhat simplistic, these independence assumptions have also been made by most previous work in the context of name disambiguation [Ferreira et al., 2010; Han et al., 2004, 2005a,b; Lee et al., 2005]. More importantly, we show, in Section 5.2, that these assumptions have little (if any) impact on the performance of different disambiguation methods, as there is little difference in their results when applied to a real (input) collection and to synthetically generated (output) collections.

The main challenge here is to infer, from the input collection, the distributions of topic popularity for each author (P^a_{Topic}), as well as the distributions of term and venue popularity associated with each topic (P^t_{Term} and P^t_{Venue}). Recall that the input collection does not contain any information on the topic(s) associated with each citation record. Thus, to address this challenge, SyGAR *models* each citation record in the input collection as a finite mixture of a set of topics. In other words, each citation record r has an associated *topic distribution*, P^r_{Topic}[3]. Terms in the work title and publication venue title are drawn from corresponding distributions associated with the topics of the citation record, and not with the authors. This model is thus able to generate a citation record with a work title containing terms (or with a venue) not used yet by any of the authors, provided that such terms (or venue) are associated with a topic of their interests.

A first step to infer P^a_{Topic}, P^t_{Term} and P^t_{Venue} consists in deriving the distribution of topics for each citation record r in the input collection, P^r_{Topic}. This is performed using the Latent Dirichlet Allocation (LDA) generative model, previously proposed for modeling document contents [Blei et al., 2003]. LDA is a three-level hierarchical Bayesian model, as illustrated in Figure 5.2. In this model, ϕ denotes a matrix of topic distributions, with a multinomial distribution of N_{Terms} terms for each of the N_{Topics} topics, which is drawn independently from a symmetric Dirichlet(α_{Term}) prior. N_{Terms} represents the total number of distinct terms in all work titles of the input

[3] $P^r_{Topic}(t)$ measures the strength at which a given topic t is related to the citation record r, normalized so as to keep the summation over all topics equal to 1. Thus, $P^r_{Topic}(t)$ can be seen as a *weight* associated with topic t for citation record r.

collection whereas N_{Topics} is the total number of topics used to model the citations. Moreover, θ is the matrix of citation-specific weights for these N_{Topics} topics, each being drawn independently from a symmetric Dirichlet(α_{Topic}) prior. For each term, z denotes the topic responsible for generating that term, drawn from the θ distribution for that citation record, and w is the term itself, drawn from the topic distribution ϕ corresponding to z. In other words, the LDA model assumes that each citation record r follows the generative process described below:

1. Draw the number of terms $size_{Title}$ in the work title according to a given distribution, such as a Poisson distribution [Blei et al., 2003] or, in our case, the distribution of number of terms in a work title for a given author a, P^a_{nTerms};

2. Draw a topic distribution θ_r for citation record r according to a Dirichlet distribution model with parameter α_{Topic}; and

3. For each term i, $i = 1 \cdots size_{Title}$, choose a topic z_i following the multinomial distribution θ_r and a term w_i from a multinomial probability conditioned on the selected topic z_i, given by distribution ϕ_{z_i}, which in turn is drawn according to a Dirichlet distribution with parameter α_{Term}.

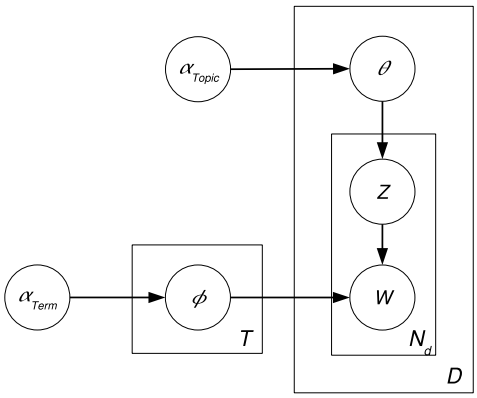


Figure 5.2. A plate representation of the LDA [Blei et al., 2003] – The LDA model assumes that each citation record r follows the generative process. r draws the number of terms N_d in the work title according to a given distribution, draws a topic distribution θ according to a Dirichlet distribution model with parameter α_{Topic} and, for each term, chooses a topic z following the multinomial distribution θ and a term w from a multinomial probability conditioned on the selected topic z, given by distribution ϕ, which in turn is drawn according to a Dirichlet distribution with parameter α_{Term}.

Thus, the LDA model has two sets of unknown parameters, namely, the topic distribution associated with each citation record r, θ_r, and the term popularity distribution of each topic j, ϕ_j, as well as the latent variables z corresponding to the

assignments of individual terms to topics. Several strategies can be adopted to estimate θ_r and ϕ_j. As in [Rosen-Zvi et al., 2004] and [Song et al., 2007], we use the Gibbs sampling algorithm [Griffiths and Steyvers, 2004]. This algorithm aims at generating a sequence of samples from the joint probability distribution of two or more random variables with the purpose of, for instance, estimating the marginal distributions of one of the variables. The Gibbs sampling algorithm constructs a Markov chain that converges to the posterior distribution of z by generating random samples from the observed data, and then uses the results to infer the marginal distributions θ_r and ϕ_j. The transitions between states of the Markov chain result from repeatedly drawing the topic of the i^{th} term, z_i, from its distribution conditioned on all other variables, that is

$$P(z_i = j | w_i = m, z_{-i}, w_{-i}) \propto \frac{C^{WT}_{m_{-ij}} + \alpha_{Term}}{\sum_{m'} C^{WT}_{m'_{-ij}} + N_{Terms}\alpha_{Term}} \frac{C^{RT}_{r_{-ij}} + \alpha_{Topic}}{\sum_{j'} C^{RT}_{r_{-ij'}} + N_{Topics}\alpha_{Topic}} \tag{5.1}$$

In other words, it computes the probability that the topic assigned to the i^{th} term (variable z_i) is j, given that the i^{th} term (variable w_i) is m and given all topic assignments not including the one related to the i^{th} term (z_{-i}). $C^{WT}_{m_{-ij}}$ is the number of times that term m is assigned to topic j excluding the current instance of term m, $C^{WT}_{m'_{-i}j}$ is the number of times that all terms in the collection are assigned to topic j excluding the current instance of m, $C^{RT}_{r_{-ij}}$ is the number of times that topic j is assigned to terms in citation record r excluding the current instance of m, $C^{RT}_{r_{-ij'}}$ is the number of times that all topics are assigned to terms in citation record r excluding the current instance of m.

From any sample from this Markov chain, we can estimate the probability of drawing a topic j for a citation r as

$$\theta_r(j) = \frac{C^{RT}_{rj} + \alpha_{Topic}}{\sum_{j'} C^{RT}_{rj'} + N_{Topics}\alpha_{Topic}}. \tag{5.2}$$

and the probability of drawing a term m for a given topic j as

$$\phi_j(m) = \frac{C^{WT}_{mj} + \alpha_{Term}}{\sum_{m'} C^{WT}_{m'j} + N_{Terms}\alpha_{Term}} \tag{5.3}$$

These distributions correspond to the predictive distributions over new terms and new topics. According to Blei et al. [2003], it is recommended to assign positive values to input parameters α_{Topic} and α_{Term} so as to allow the selection of new topics and new terms that have not been previously observed. In other words, positive values for

these parameters ultimately imply in non-zero probabilities to all items (i.e., topics or terms) regardless of whether they have C_{rj}^{RT} (or C_{mj}^{WT}) equal to 0.

SyGAR follows the aforementioned procedure by processing all citation records in the input collection, one at a time. It uses the terms in the work titles to estimate the conditional probability given by Equation 1. After a number of iterations, it estimates the topic distribution of each citation record r, P_{Topic}^{r} (given by θ_r in Equation 5.2) and the term popularity distribution per topic t, P_{Term}^{t} (given by ϕ_j in Equation 5.3)

Afterwards, the tool infers the topic distribution P_{Topic}^{a} of each author a by combining the weights of the topics of all citation records in which a is an author. Only topics with weights greater than or equal to β_{Topic} (input parameter) are selected from each citation record of a, so as to avoid introducing topics of very little interest to a in P_{Topic}^{a}. SyGAR also infers the venue popularity distribution of each topic t, P_{Venue}^{t}, by combining the weights of t associated with citation records containing the same publication venue, provided that t has the largest weight among all topics of the given citation record, i.e., provided that t is the most related topic of the given citation record[4].

Given the author profiles, SyGAR is ready to generate the synthetic citation records. It generates a number N_{loads} of batches of data representing a number of successive loads. For each load, it generates a number of records given by N_R or, alternatively, specified based on the distributions of the number of publications per author per load (as in Section 5.3.2).

Since SyGAR extracts publication profiles of all authors in the input collection, the term "author" was used up to this point in this section to refer to any author in the input collection, regardless of whether she has an ambiguous name or not. Since our present goal is to evaluate disambiguation methods, we here use SyGAR to generate synthetic records only for authors *with ambiguous names*. Thus, for the sake of clarity, through the rest of this chapter and unless otherwise noted, we refer to authors *with ambiguous names*, the main target of our study, as simply *authors*, treating all other authors in the input collection as their *coauthors*.

The following three sections describe how SyGAR generates synthetic records for authors (with ambiguous names) already present in the input collection (Section 5.1.2) and for new authors (Section 5.1.3), as well as how it models dynamic publication profiles (Section 5.1.4) and how it modifies citation records in its final step (Section 5.1.5).

[4]These probabilities are combined by first summing up all values of $C_{rj}^{RT} + \alpha_{Topic}$ (numerator in Equation 2) for citations r and topics j of interest, and then normalizing them so as to keep the total probability equal to 1.

5.1.2 Generating Records for Existing Authors

Each synthetic record for existing authors is created as follows:

1. Select one of the authors of the collection according to the desired distribution of number of records per author. Let it be a.
2. Select the number of coauthors according to $P^{a}_{nCoauthors}$. Let it be a_c.
3. Repeat a_c times:
   - with probability 1 - $\alpha_{NewCoauthor}$, select one coauthor according to $P^{a}_{Coauthor}$;
   - otherwise, uniformly select a *new coauthor* among remaining coauthors in the input collection.
4. Combine the topic distributions of a and each of the selected coauthors. Let it be P^{all}_{Topic}.
5. Select the number of terms in the title according to P^{a}_{nTerms}. Let it be a_t.
6. Repeat a_t times: select one topic t according to P^{all}_{Topic} and select one term for the work title according to P^{t}_{Term}.
7. Select the publication venue:
   - with probability 1 - $\alpha_{NewVenue}$, select a venue according to P^{t}_{Venue}, where t is the topic that was selected most often in Step 6;
   - otherwise, randomly select a *new venue* among remaining venues in the input collection.

Step 1 uses either the collection profile, i.e., $P^{c}_{nRecordsAuthors}$, or a distribution specified as part of the input. The latter may be specified by, for instance, providing the fractions of records to be generated for each author. This alternative input adds flexibility to our tool as it allows one to experiment with various scenarios by generating synthetic collections with varying numbers of records per author profile. Steps 2 and 5 use the distributions in the profile of the selected author. The same holds for Steps 3 and 7, although, with probabilities $\alpha_{NewCoauthor}$ and $\alpha_{NewVenue}$, SyGAR selects new coauthors and new venues (i.e., coauthors and venues that are not associated with the selected author in the input collection), respectively. We also note that, in Steps 3 and 6, we do not allow for a coauthor (or term) to be selected more than once.

The combined topic distribution P^{all}_{Topic} (Step 4) is obtained by first selecting only the topics that are shared by *all selected authors* (a and her coauthors). If there is

no shared topic, we take the union of all topics associated with the selected author a and the coauthors. The combined distribution is built by, for each topic t, averaging $P^a_{Topic}(t)$ across all authors (a and the coauthors) and normalizing these values at the end so as to keep the summation over all topics equal to 1.

The seven steps are repeated a number of times equal to the target number of records in the new load.

5.1.3 Adding New Authors

Another use for SyGAR is to generate records for large author populations, by building citation records not only for the authors present in the input collection but also for new (non-existing) authors. A variety of mechanisms could be designed to build such records. For the sake of demonstrating SyGAR's flexibility, we here adopt a strategy that exploits the publication profiles from author and co-authors, extracted from the input collection. Other (possibly more sophisticated) approaches will be designed in the future.

A new author a is created by first selecting one of its coauthors among all authors (with ambiguous and non-ambiguous names) in the input collection, i.e., using $P^c_{nRecordsAllAuthors}$. Let say it is c_a. The new author inherits c_a's profile, but the inherited topic and coauthor distributions are changed as follows. First, the new author inherits only a percentage $\%_{InheritedTopics}$ of the topics associated with c_a, i.e., the topics that are more strongly related to her (i.e., with largest weights). Let l_{Topic} be the list of inherited topics. The new author's topic popularity distribution is built by using the same weights c_a's distribution for the inherited topics, rescaling them afterwards so as to keep the summation equal to 1.

Similarly, we set a's coauthor list equal to c_a plus all coauthors of c_a that have at least one of the topics in l_{Topic} associated with them. Once again, the probabilities of selecting each coauthor are also inherited, and rescaled afterwards. However, we force that c_a appears in all records generated to the new author. This strategy mimics the case of a new author who, starting its publication career, follows part of the interests (topics) of one who will be a frequent coauthor (e.g., advisor or colleague).

Finally, the name of the new author is generated with the initial of the first name and the full last name of an existing author (i.e., an ambiguous author name), selected from the input collection using the distribution of the number of records per ambiguous group, i.e., $P^c_{nRecordsGroup}$.

Parameter $\%_{NewAuthors}$ specifies the percentage of new authors generated for each new load.

5.1.4 Changing an Author's Profile

SyGAR also allows one to experiment with dynamic author profiles, mimicking scenarios in which authors (with ambiguous names) may change their publication profiles over time due to shifts in interests, as in the real-world bibliographic digital libraries. Although SyGAR processes the input collection as a static snapshot of publication profiles, the tool can generate collections in which authors dynamically change their attribute distributions over successive loads.

In the lack of a previous characterization of dynamic properties of author publication, SyGAR currently implements a simple strategy to change the *topic distribution* of an author a (illustrated in Figure 5.3-a). It first sorts the topics associated with a according to their probabilities (i.e., P^a_{Topic}) so as to have a histogram as close to a bell shape as possible (i.e., mode in the center and least probable topics in both extremes), as illustrated in Figure 5.3-b. It then shifts the distribution along the x-axis by a factor of δ, at each load. Figure 5.3-c illustrates four successive changes in an author's profile using δ equals to 5.

By carefully choosing δ, this procedure guarantees that changes occur as softly as desired, mimicking the case of an author smoothly increasing/decreasing her interest in some topics over time.

Parameter $\%_{ProfileChanges}$ specifies the percentage of authors that will experience changes in their profiles in each load.

5.1.5 Modifying Citation Attributes

The final step in the citation record generation process consists of modifying the citation attributes according to several input probability distributions (see Table 5.1). Two mandatory changes refer to how an author's first and middle names should be presented in the citation record. There are three possibilities: completely remove the first/middle name, keep the first/middle name entirely and keep only the initial of the first/middle name. Probability distributions P^{FName} and P^{MName} are used to make the selections, which are applied to the names of all authors and coauthors in the synthetic citations.

Next, six input distributions may be used to introduce typographical errors in the generated records. $P^{LName}_{\#Mods}$, $P^{Title}_{\#Mods}$ and $P^{Venue}_{\#Mods}$ are used to draw the number of modifications in each author's last name, work title and publication venue, respectively, whereas P^{LName}_{Mod}, P^{Title}_{Mod} and P^{Venue}_{Mod} are used to draw the type of each such modification in each attribute. Four modifications are possible, namely, insert, remove or change one character and swap two randomly selected characters.

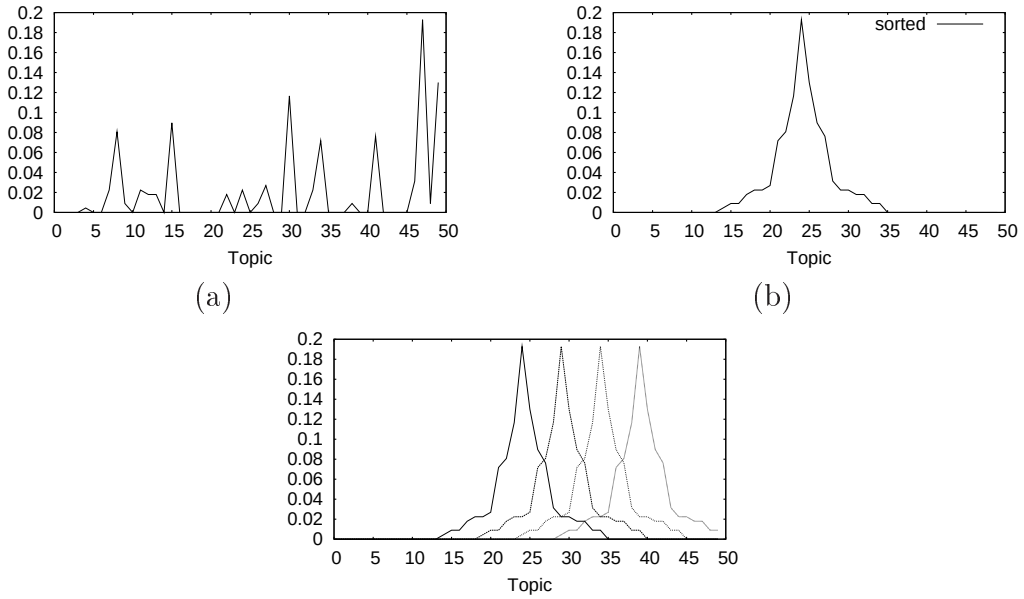


Figure 5.3. Changing author a's profile by altering her topic distribution. (a) the original topic distribution of author a. (b) The topics associated with a sorted according to their probabilities (P^a_{Topic}) so as to have a histogram as close to a bell shape as possible. (c) The topic distribution shifted along the x-axis by a factor $\delta = 5$; 2 shifts are shown in the figure.

In its current version, SyGAR allows for the easy experimentation with a multitude of relevant scenarios (see examples in Section 5.3) that occur in real digital libraries. We intend, in the future, to design even more sophisticated mechanisms to add new authors to the output collection as well as new strategies to introduce changes in the profiles of existing authors and in the synthetic records.

As a final note, we emphasize that, although SyGAR was designed to help addressing the name disambiguation task, it can be used to generate any collection of citation records, as long as a real collection is provided as source of author profiles. Thus, we believe it can be used to study other problems related to bibliographic digital libraries as well (e.g., scalability issues). SyGAR is implemented in Java and will be available for public use in due time.

5.2 Validation

As the methods available in the literature adopt a variety of solutions, including unsupervised and supervised techniques (see Section 3), we here select three methods, each one representative of a different technique: the SVM-based name disambiguation

method (SVM) that is proposed in [Han et al., 2004], unsupervised Heuristic-based Hierarchical Clustering method (HHC) that is proposed in [Cota et al., 2010] and the K-way Spectral Clustering-based method (KWAY) that is proposed in [Han et al., 2005b].

We validate SyGAR by comparing the performance of the selected name disambiguation methods on real and synthetically generated collections as well as by comparing attribute distributions (author/coauthor and topic distributions) of both collections. The real collection of citation records used in our study was that extracted from DBLP (see Section 2.4).

The ultimate goal of a disambiguation method is to separate all records within each ambiguous group into a number of subgroups (i.e., clusters), one for each different (disambiguated) author. In order to evaluate the performance of the disambiguation methods, we here use the K metric that for us better evaluate the disambiguation results.

Since our main goal is to use SyGAR to evaluate representative disambiguation methods, our main validation consists of assessing whether the synthetically generated collection captures the aspects of the corresponding real collection (and its ambiguous groups) that are of relevance to the disambiguation methods. Towards that goal, we compare the K result obtained when each of the three selected disambiguation methods is applied to the real collection and its corresponding synthetic versions.

In our validation experiments, we set α_{Topic}=α_{Term}=0.00001, thus allowing, with a very small probability, the selection of any topic/term, regardless of whether they were associated with the selected authors/topics in the input collection (see Section 5.1.1). We believe this leads to the generation of more realistic synthetic collections. Moreover, we set β_{Topic}=0.07, i.e., to infer the topic distribution of each author, we combine the topics with weights greater than or equal to 0.07 in each citation record of such author, avoiding introducing topics with very little interest to her in the topic distribution. The number of authors and the number of records per author in the synthetic collections are set to be the same as in the input collection, as both parameters have impact on the effectiveness of the methods, and thus should be kept fixed for validation purposes. In other words, we let N_R and $P^c_{nRecordsAuthors}$ be the same as in the input collection and make N_{loads}=1. For validation purposes, we set $\%_{NewAuthors} = \%_{InheritedTopics} = \%_{ProfileChanges} = \alpha_{NewCoauthor} = \alpha_{NewVenue} = \delta = 0$, keep first and middle names of each author as in the input collection and avoid introducing any typographical error in the synthetic collections. We experiment with N_{Topics} equal to 300 and 600. We further discuss issues related to the sensitivity of SyGAR to these parameters later in this section.

Regarding the parameters for the methods, for SVM, we used the implementation provided by the LibSVM package [Chang and Lin, 2001], with RBF (Radial Basis Function) as the kernel function, where the best γ and cost values were obtained from the training data using the *Grid* program, available with the LibSVM package. For KWAY, we used the implementation of the K-way spectral clustering provided by the University of Washington spectral clustering working group[5] and the number of authors in the collections as the target number of clusters to be generated. For HHC, we used the same values specified in [Cota et al., 2010] for the work and venue title similarity thresholds.

For the sake of evaluation, we divided the real collection as well as each synthetic collection generated from it into two equal-sized portions, by randomly splitting the author records into two parts. One is the training data and the other is the test set. We then applied each method to each ambiguous group in the test set. The supervised method uses the training data to learn the disambiguation model. We repeated this process 10 times for each collection, presenting results that are averages of the 10 runs.

Table 5.2. SyGAR validation – Average K results and 95% confidence intervals for real and synthetically generated collections (N_{Topics} = 300). Statistical ties are in bold.

Collection	KWAY	SVM	HHC
Real	0.530±0.009	0.764±0.005	**0.770**±0.006
Synthetic 1	0.478±0.005	0.698±0.008	**0.753**±0.013
Synthetic 2	0.484±0.007	0.706±0.005	0.750±0.011
Synthetic 3	0.478±0.008	0.701±0.006	0.752±0.005
Synthetic 4	0.480±0.006	0.708±0.007	0.755±0.006
Synthetic 5	0.477±0.009	0.702±0.006	0.751±0.011

Table 5.2 shows average K results, along with corresponding 95% confidence intervals, for the three disambiguation methods applied to the real collection and to five synthetically generated collections[6] using N_{Topics}=300. Note that the synthetic collections are only slightly more difficult to disambiguate than the real one. Indeed, K results for KWAY, SVM and HHC methods are, on average, around only 17%, 11% and 2.3%, respectively, smaller in the synthetic collections, including a statistical tie between the real and a synthetic collection using the HHC method (marked in bold). We notice that, the number of distinct terms in the work titles used by each author in the synthetic collection with N_{Topics}=300 is around 9% greater than in the real collection. Since KWAY relies directly on the similarity among the records to group

[5] *http://www.stat.washington.edu/spectral*

[6] These collections were built based on the same input parameters, differing only with respect to the seed used in the random number generator.

them, which uses the work title, this may explain the larger difference for this method. HHC, on the other hand, first groups by coauthor and only uses the information in the work and publication titles for minimizing the fragmentation problem, while SVM, relies on the training data, being more robust to these changes. This may explain the smaller differences between these methods when applied to the synthetic and real collections.

We consider these results very good, given the complexity of the data generation process, and considering that SyGAR allows for the selection of title terms and venues not previously associated with an author. In other words, the synthetic collections, built using SyGAR, are mimicking reasonably well the real data.

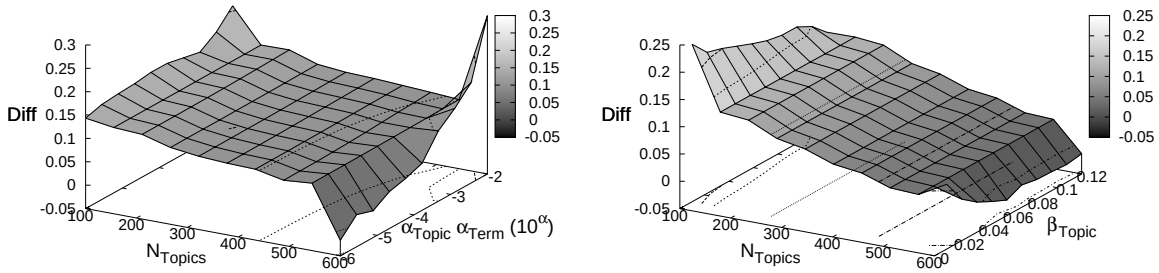
Table 5.3. SyGAR Validation: Average K results and 95% confidence intervals for real and 5 synthetically generated collections ($N_{Topics} = 600$).

Collection	KWAY	SVM	HHC
Real	0.530±0.009	0.764±0.005	0.770±0.006
Synthetic 1	0.499±0.008	0.746±0.007	0.793±0.008
Synthetic 2	0.489±0.006	0.743±0.007	0.790±0.009
Synthetic 3	0.493±0.006	0.742±0.007	0.799±0.012
Synthetic 4	0.491±0.006	0.750±0.006	0.796±0.006
Synthetic 5	0.497±0.010	0.743±0.010	0.801±0.008

Table 5.3 shows similar results for synthetic collections built using N_{Topics}=600. Note that these collections are easier to disambiguate and the K results are closer to those produced for the real collection. Indeed, comparing real and synthetic collections, results for KWAY and SVM are, on average, only 9% and 4% smaller in the synthetic collections, whereas the HHC results are slightly better in the synthetic collections (3.3%, on average). These results further show that SyGAR is capable of capturing the aspects of the real collection that are relevant to the disambiguation methods.

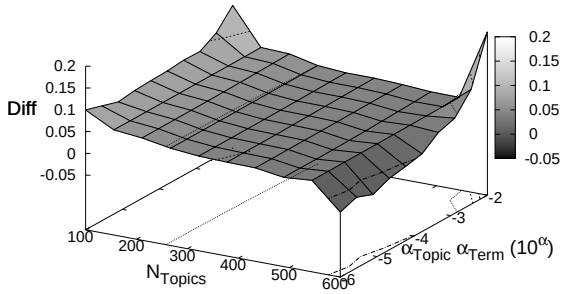
The reason why using 600 topics instead of 300 leads to synthetic collections on which the disambiguation methods produce results closer (or even slightly better) to the results for the corresponding real collections may be explained as follows. As the number of topics increases, the number of authors sharing any given topic tends to decrease. As a consequence, when building a synthetic citation, there is a higher chance that a term selected for a given topic (Equation 5.3) has been actually used, in the real collection, by the author to which that topic was associated. Recall that, when generating a citation, if the selected authors share no topic, SyGAR combines all topics of individual authors. This happens with the majority of the citations generated when we set N_{Topics}=600. Thus, in this case, the chance of generating synthetic citations with terms that were used by at least one of the authors in the real collection is higher,

which ultimately makes the synthetic citations look more similar to the real ones, at least with respect to title terms. This leads to synthetic collections that better resemble the real ones, therefore justifying the similar performance of the methods.

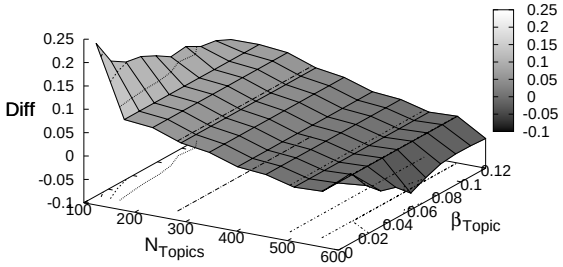


(a) SVM

(b) SVM



(c) HHC



(d) HHC

Figure 5.4. Sensitivity of SyGAR to α_{Topic}, α_{Term}, β_{Topic} and N_{Topics} – Relative error between performance of each method on synthetic and real collections. (a) and (c) show the results of SVM and HHC, respectively, when applied to synthetically generated collections using various values of α_{Topic}, α_{Term} and N_{Topics}, keeping $\beta_{Topic} = 0.07$. (b) and (d) show the results of SVM and HHC, respectively, when applied to synthetically generated collections using various of β_{Topic} and N_{Topic}, keeping α_{Topic}=α_{Term}=10^{-5}.

To better understand the sensitivity of SyGAR to some of its key parameters, we evaluate the results of two of the selected methods, namely SVM and HHC, when applied to synthetic collections generated using various values of α_{Topic}, α_{Term}, β_{Topic}, and N_{Topics}. We report, for each method, the relative difference of its performance on the real and synthetic collections, here referred to as the *relative error*. A positive error implies that the synthetic collection is harder to disambiguate than the real one. We report average results of **five** runs, omitting confidence intervals for the sake of clarity.

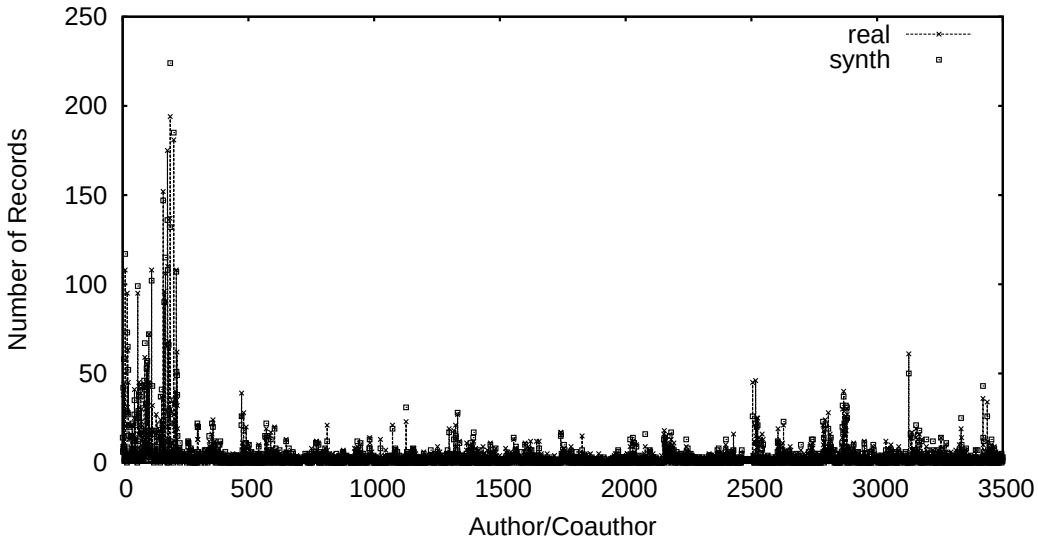
We start by showing, in Figures 5.4(a) and 5.4(c), average errors for SVM and HHC, respectively, as we vary α_{Topic} and α_{Term} from 10^{-6} to 10^{-2} (setting both to the same value in each case), and N_{Topics} from 100 to 600, while keeping β_{Topic} fixed at 0.07. We note that, as both α_{Topic} and α_{Term} increase, the gap between the results on synthetic and real collections tends to increase significantly for both methods, particularly for large number of topics. The synthetic collections become harder to disambiguate for larger values of α_{Topic} and α_{Term}. This is because larger values of both parameters impact the computation of P^r_{Topic} and P^t_{Term} (Equations 5.2 and 5.3, respectively) more significantly. This is particularly true if N_{Topics} is large, since counters C^{RT}_{rj} and C^{WT}_{mj}, inferred from the input collection, tend to decrease as the number of topics increases. In other words, larger values of α_{Topic} and α_{Term} may introduce too much noise in the estimates of P^r_{Topic} and P^t_{Term}, ultimately generating synthetic collections that are much harder to disambiguate than the real collection. The same can be noticed, though to a less extent, for smaller number of topics.

Moreover, the errors also tend to decrease as the number of topics (N_{Topics}) increases, provided that the values of α_{Topic} and α_{Term} are not very large. As previously discussed, the larger the number of topics, the higher the chance of generating citations with terms that were used by at least one of its authors in the real collection. One extreme case is N_{topics}=600 and $\alpha_{Topic} = \alpha_{Term}$=$10^{-5}$, when, as previously discussed, this happens to most generated citations and both methods produce results that are very close to those obtained with the real collection. Thus, we suggest to use α_{Term}=10^{-5}, α_{Topic}=10^{-5} and N_{Topics}=600.

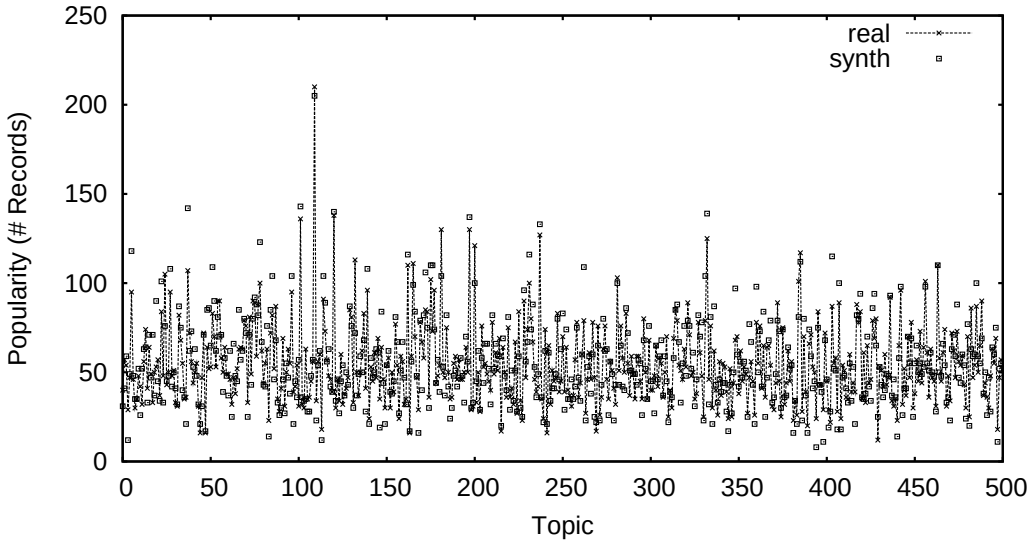
Next, Figures 5.4(b) and (d) show average errors for SVM and HHC, respectively, as we vary β_{Topic} from 0.01 to 0.12 and N_{Topics} from 100 to 600, keeping α_{Topic}=α_{Term}=10^{-5}. In general, both methods tend to produce results closer to those obtained with the real collection for larger values of β_{Topic}. This is expected as β_{Topic} represents the minimum weight of topics that can be associated with an author. Thus, in general, larger values of β_{Topic} tend to reduce the chance of associating to an author a topic that is of little interest to her. So, the β_{Topic} value must be lower or equal to 0.10. Therefore, we suggest to set β_{Topic}=0.10.

We further validate SyGAR by comparing some of the attribute distributions in the real and synthetic collections. As a sanity check, Figure 5.5(a) shows the distributions of the number of records per author/coauthor ($P^c_{nRecordsAllAuthors}$) in the real and in a synthetically generated collection. Clearly, both distributions are very similar, as expected, with the average absolute difference between real and synthetic collection around 1.29. Figure 5.5(b), in turn, shows the popularity (in terms of number of citations) of topics, in the real and in a synthetic collection built using N_{Topics}=600,

with the average absolute difference between real and synthetic collection around 8.63. Recall that this metric is *not* directly manipulated by SyGAR. Once again, the curves show very similar patterns. Similar agreement was also obtained for collections generated using other values of N_{Topics} as well as for other attribute distributions.



(a) Number of Records per Author/Coauthor



(b) Topic Popularity (N_{Topics}=500)

Figure 5.5. SyGAR validation. We use α_{Topic}=α_{Term}=10^{-5} and β_{Topic}=0.7.

5.3 Evaluation of Disambiguation Methods with SyGAR

We demonstrate the applicability of SyGAR by evaluating the SVM, KWAY and HHC disambiguation methods in three realistic scenarios generated by our tool. We start by describing these scenarios in Section 5.3.1. We then present our experimental setup in Section 5.3.2 and discuss the most relevant results from our evaluation in Section 5.3.3. We emphasize that our goal here is *not* to thoroughly evaluate the selected methods but rather to show how our tool can be used to evaluate existing methods in relevant realistic situations.

5.3.1 Analysis Scenarios

We envision three scenarios that capture some relevant dynamic patterns observed in real digital libraries. All three scenarios encompass a live digital library (DL) evolving over the period of several years. In its initial state, the DL is a collection of synthetic citations. At the end of each year, a load is performed into the DL with new citations of existing, and, possibly, of new ambiguous authors, depending on the specific scenario. We choose to model yearly loads, using as parameters the average yearly publication rates of authors in each ambiguous group, extracted from DBLP (see Section 5.3.2). However, the load period could be easily changed.

Scenario 1 consists of an evolving digital library with new citations introduced at each new load, assuming a fixed author population with static publication profiles. In other words, Scenario 1 captures solely the impact of an evolving DL. Only authors (with ambiguous names) in the original input collection are considered and they *do not* change their profiles during successive loads, keeping their topic and coauthor distributions as extracted from the input collection.

Scenario 2 considers the introduction of new authors to the existing author population. New authors are added to the collection at a given rate in each successive load. As described in Section 5.1.3, a new author inherits a percentage of the topics of an author that will be considered as her main coauthor (e.g., an advisor). Moreover, all publications of a new author have her main coauthor in the author list.

Finally, *Scenario 3* considers authors with dynamic profiles. A percentage of the current authors make small changes in their profiles before each new load, i.e., their topic distributions are shifted by a factor δ, as explained in Section 5.1.4. The changes are very small, but are performed at a constant rate over the years. Although this might not be very realistic, it allows us to test the limits of the disambiguation

methods under dynamic publication profiles. As we are unaware of previous studies measuring profile change rates in real-word digital libraries, any choice of rate would be arbitrary.

Thus, the envisioned scenarios allow us to evaluate the robustness of the selected disambiguation methods to three key real-world aspects: (1) the evolution of the DL, (2) the inclusion of new authors with ambiguous names into the DL and (3) changes in author profiles. We emphasize that these are only a few of the scenarios that can be generated using SyGAR. For instance, scenarios with different, possibly heterogeneous, profile change rates, i.e., different values of δ for different authors, can also be devised, being the loads easily produced by SyGAR. Building and experimenting with other scenarios is subject of future work.

5.3.2 Experimental Setup

We performed experiments with the same collection used in Section 5.2, containing 11 ambiguous groups, as shown in Table 2.3. For each scenario, the number of synthetic citation records in the initial state s_0 of the digital library is the same as in the real collection. Ten successive data loads, one per year, are generated using SyGAR (i.e., $N_{loads} = 10$), parameterized by the real collection as source of publication profiles as well as with additional inputs according to the specific scenario.

Starting at state s_i, the new citation records generated by SyGAR are disambiguated using each one of the three methods and the results are incorporated into the corresponding DL version, which evolves into state s_{i+1}. If the supervised SVM method is used, SyGAR is also used to generate a training set containing the same number of citations of the DL at its initial state s_0. This training set is used by SVM to "learn" its model to disambiguate the records generated at each load. For both KWAY and HHC methods, the generated records are first incorporated into the current state of the DL and the disambiguation is performed with all records.

For each new load, SyGAR generates records for authors already in the DL and, in Scenario 2, for new authors. The synthetic citations are generated using N_{Topics}=600, β_{Topic}=0.10 and α_{Topic}=α_{Term}=10^{-5}. Moreover, in all three scenarios, we set $\alpha_{NewVenue}$=$\alpha_{NewCoauthor}$=0, thus restricting the selection of venues and coauthors for an author's new citation to those already associated with her in the input collection.

We also format author and coauthor names according to probabilities p that match the observed patterns in the input collection. In particular, we retain either only the initial of the first name (p=0.53) or the complete first name (p=0.47). Moreover,

regarding the middle name, we either keep only the initial (p=0.37), remove it (p=0.53) or keep it completely (p=0.10). Finally, we introduce no typographical errors in any attribute.

For experiments with Scenario 2, the number of new authors to be added at each new load is specified as a fraction $\%_{NewAuthors}$ of the total number of authors in the DL at its current state. We experiment with values of $\%_{NewAuthors}$ equal to 5% and 10%. Each new author inherits 80% of the topics associated with her most frequent coauthor ($\%_{InheritedTopics}$=80%). We note that newly added authors remain as part of the DL throughout the rest of the experiment, i.e., records are generated for these authors in all successive loads.

Moreover, for experiments with Scenario 3, changes are introduced in a percentage $\%_{ProfileChanges}$ of author profiles across successive loads using a shift δ=5. We experimented with $\%_{ProfileChanges}$ equal to 10%, 50% and 100%. In this case, in each yearly load a different set of authors from the previous state is chosen to have their profiles changed.

Finally, the distribution of the number of records generated for each author is built from the data presented in Table 5.4, which shows the distribution of the average number of publications per year per (existing and new) author. These distributions were extracted from DBLP, counting the number of publications of each author of three selected ambiguous groups during the period of 1984-2008. We selected groups “C. Chen", “A. Gupta" and “D. Johnson" which, as shown in Table 2.3, have very different author population sizes. “C. Chen" is a very large ambiguous group with 60 different authors. “D. Johnson", on the contrary, is much smaller, and “A. Gupta" has an intermediary number of authors.

Table 5.4. Distribution of average number of publications per year per author (DBLP: 1984 - 2008).

	Average Number of Publications per Year			
	One	Two	Three	> Four
New Authors	55%	30%	10%	5%
Existing Authors	14%	42%	28%	16%

For loads s_1 to s_{10}, the generation of new records use the distributions shown in Table 5.4. We chose to use that distribution because Han et al.'s DBLP collection, which we use here, did not have temporal information, so the number of records per author ($P^c_{nRecordsAuthors}$) is a cumulative measure, and using it would certainly generate distortions depending on the length of the career of that author. For generating the successive loads, the yearly rates of publication are more important.

5.3.3 Evaluation of Results

The following subsections present our evaluation of the three selected methods in each considered scenario built using SyGAR. Our evaluation is carried out by computing the K value at each state of the DL. The results reported in the following sections are averages of five runs. Corresponding 95% confidence intervals are usually very tight, indicating errors on the reported means that fall below 12% in all cases.

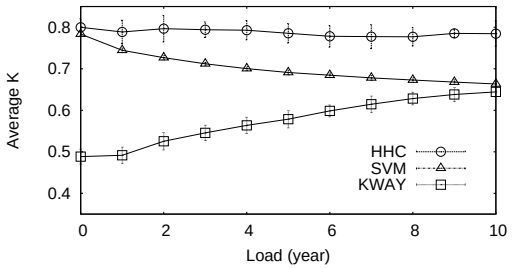


Figure 5.6. Scenario 1 – Evolving DL with static author population and publication profiles.

5.3.3.1 Scenario 1: Evolving DL with Static Author Population and Profiles

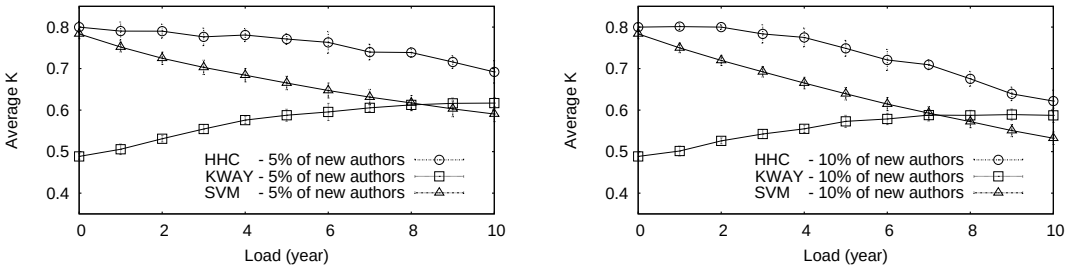
Figure 5.6 shows, for each disambiguation method, the average K value computed over all 11 ambiguous groups in each state of the digital library over the ten-year period. Corresponding 95% confidence intervals are also shown. Note that the relative order of the methods, in terms of achieved performance, remains the same through all states: HHC outperforms SVM, which, in turn, outperforms KWAY. However, the three methods have very different behaviors as new loads of citations are introduced into the DL.

SVM's performance, for example, tends to decrease over time: while it starts in the first load (s_0) with an average K value equals to 0.78, these values fall to levels around 0.66 in the successive loads. Indeed after 10 loads, SVM's performance degrades by 15%. This degradation is possibly due to errors caused by imprecise models learned for authors with very few records in the training set. These errors are cumulative in the successive loads, calling for a retraining of SVM. Analyzing SVM with retraining is not an easy task as factors such as errors introduced in the collection may affect the results of these experiments. Thus, we leave it for future work.

KWAY, on the contrary, experiences an increasing improvement in effectiveness as new citations are added. This occurs because there is incrementally more information about each author, helping KWAY to better characterize them. Indeed, the gain in

performance after 10 loads reaches 32%. Unlike both SVM and KWAY, HHC remains with approximately the same performance, varying by at most 2%, throughout all 10 successive loads. This is possibly due to the specific heuristics exploited by HHC for the name disambiguation task (see [Cota et al., 2010] for details), in contrast to the general purpose techniques used by SVM and KWAY.

As consequence of such distinct behaviors, we find that, while in the beginning (i.e., state s_0) HHC outperforms SVM by only 2% (on average) and SVM outperforms KWAY by 60% (on average), corresponding performance gains switch to 18% and only 3%, respectively, after the last load of new citations.



(a) $\%_{NewAuthors}$=5% (b) $\%_{NewAuthors}$=10%

Figure 5.7. Scenario 2 – Evolving DL and addition of new authors ($\%_{InheritedTopics}$=80%).

5.3.3.2 Scenario 2: Introduction of New Authors

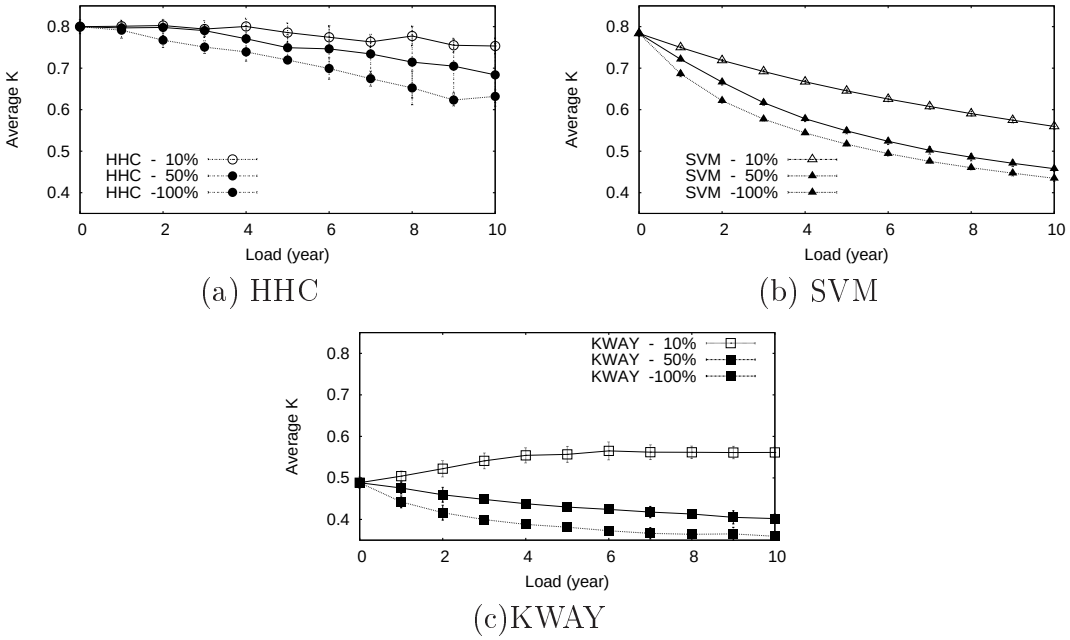
We now use SyGAR to analyze the impact on each method of introducing new authors to the current author population. Figure 5.7 shows average K values and corresponding 95% confidence intervals for each method on collections built using $\%_{NewAuthors}$ equal to 5% and 10%, and $\%_{InheritedTopics}$ equal to 80%.

The behaviors of both KWAY and SVM follow trends very similar to those observed in Figure 5.6: whereas SVM suffers performance degradation, KWAY actually improves in performance as new loads of citations are added to the DL. However, we note a clear detrimental impact of the introduction of new authors on both methods. SVM's performance degrades much faster for $\%_{NewAuthors}$=10% than for $\%_{NewAuthors}$=5%. Indeed, in comparison with the case of static author population (Figure 5.6), the average K values after the last load are 20% and 11% worse for $\%_{NewAuthors}$ equal to 10% and 5%, respectively. Recall that SVM uses the same training set, containing only records of the existing authors in state s_0, to disambiguate the DL in all states. Therefore, SVM is unable to recognize new authors, thus introduc-

ing errors into the DL when disambiguating their records. Once again, SVM requires retraining when facing the addition of new authors to the DL, a subject of future study.

Similarly, the improvement in performance experienced by KWAY becomes less significant as the fraction of new authors introduced at each load increases. This happens because of the increase in the number of authors, which implies in higher ambiguity and a higher inherent difficulty in distinguishing them. In comparison with the case reported in Figure 5.6, KWAY's performance after the last load is 9% and 4% worse for $\%_{NewAuthors}$ equal to 10% and 5%, respectively. In fact, for both values of $\%_{NewAuthors}$, KWAY outperforms SVM after the last load.

Figure 5.7 also shows that, like SVM and KWAY, HHC also suffers a significant performance degradation with the introduction of new authors. Indeed, for $\%_{NewAuthors}$=10%, the difference in average performance between HHC and KWAY drops from 64% to only 6% after the last load. In comparison with the case of static author population, average K values after the last load are 21% and 12% worse for $\%_{NewAuthors}$ equal to 10% and 5%, respectively.



(a) HHC (b) SVM

(c)KWAY

Figure 5.8. Scenario 3 – Dynamic author profiles ($\delta = 5$ and $\%_{ProfileChanges}$=10%, 50% and 100%).

5.3.3.3 Scenario 3: Dynamic Author Profiles

Finally, Figures 5.8(a-c) show average K values and corresponding 95% confidence intervals when a fraction $\%_{ProfileChanges}$ equal to 10%, 50% and 100% of the authors

experience changes in their profiles at each new load. All three methods greatly suffer if facing dynamic changes in profiles. KWAY, in particular, which experiences performance improvements in both Scenarios 1 and 2, now suffers some degradation for values of $\%_{ProfileChanges}$ greater than or equal to 50%. In particular, taking Scenario 1 and the performance of each method after the last load as basis for comparison, we note that SVM's performance degrades by 16%, 31% and 34% for $\%_{ProfileChanges}$ equal to 10%, 50% and 100%, respectively. HHC, in turn, experiences a performance degradation of 4%, 13% and 19% in the respective cases, being therefore more robust than SVM in this scenario. KWAY, which seems very robust to Scenarios 1 and 2, still experiences some performance improvement (by as much as 14%) if $\%_{ProfileChanges}$ is equal to only 10%. Notice however, that this improvement is smaller than in the scenarios in which we did not have profile changes (in that case, improvements went up to 32%). However, for values of $\%_{ProfileChanges}$ equal to 50% and 100%, its performance degrades by 18% and 26%, respectively. While KWAY was able to take advantage of the increase in information in Scenarios 1 and 2, the change in the profile of existing authors confounds this method.

In sum, the performance of SVM tends to degrade over time, particularly as new authors are introduced in the collection. In contrast, the performance of the unsupervised KWAY method, which uses privileged information regarding the number of authors in the digital library, tends to increase with time, except when there are changes in the author profiles. Overall, among the three methods, the heuristic-based method HHC, designed specifically to address the name disambiguation problem, has the best performance in the analyzed situations.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we presented a set of contributions to help solving the author name ambiguity problem. First of all, we presented a taxonomy to classify the author name disambiguation methods that helps better understand how the methods work and consequently understand their limitations. Our taxonomy classifies the disambiguation methods according to the type of approach, such as author grouping methods that group the references to the same author using the similarity among the reference attributes, and author assignment methods that assign the references to their authors, or according to the evidence explored in the disambiguation task, for instance, methods that use citation attributes, Web information or implicit attributes. Further, we described several automatic representative disambiguation methods using the taxonomy.

Second, we proposed SAND, a new hybrid disambiguation method that exploits the strengths of both supervised author assignment and unsupervised author grouping methods. In its first step (i.e., the author grouping step), the references are clustered so that references that are likely to be associated with the same author are grouped together in clusters. In its second step (i.e., the cluster selection step), some of these clusters are selected to be used as training data. In its third step (i.e., the author assignment 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 extracted from the DBLP and BDBComp digital libraries to evaluate SAND. In the DBLP collection, SAND outperformed two unsupervised methods by more than 27%. In the BDBComp collection, SAND outperformed two unsupervised methods by more than 36% under the pF1 metric and by more than 4% under the K metric. SAND

also demonstrated to be very competitive, sometimes superior, to supervised author assignment methods. In our evaluation, we also showed that without any parameter setup SAND produces the best result.

And, finally, we proposed SyGAR, a new generator of synthetic citation records that helps to evaluate author name disambiguation methods under several scenarios. SyGAR generates synthetic citation records following the publication profiles of existing authors, extracted from the input collection. Moreover, SyGAR allows the simulation of several real-world scenarios, such as the introduction of new authors (not present in the input collection) and dynamic changes in an author's publication profile, as well as the introduction of typographical errors in the synthetic citations (not addressed here). We validated it by comparing the results produced by three representative disambiguation methods on a standard real collection and on synthetic collections produced using our tool. The selected methods are: the supervised SVM-based method, the heuristic HHC method and the unsupervised KWAY clustering-based method. Our validation experiments show a very good agreement in the performance obtained for all three methods for real and synthetically generated collections. We further analyzed SyGAR by demonstrating its applicability to evaluate the selected methods under three real-world scenarios, namely the evolution of a DL with static author population and publication profiles, the introduction of new authors and the dynamic changes in the author's profiles. Our results indicate that the performance of SVM tends to degrade with time, particularly as new authors are introduced in the collection. In contrast, the performance of the unsupervised KWAY method, which uses privileged information regarding the number of authors in the digital library, tends to increase with time, except when there are changes in the author's profiles. Overall, among the three methods, the heuristic HHC method, designed specifically to address the name disambiguation problem, has the best performance in all analyzed scenarios.

6.2 Future Research

Regarding SAND, several aspects may be further investigated:

- Other manners to identify when a reference belongs to an author who does not have any citation record in the digital library instead of just using the number of association rules projected from the training data to decide whether a new reference belong to a new author or not.
- Situations in which only the first step – author grouping step – is sufficient to disambiguate an ambiguous group. There are collections, such as BDBComp, in

which the author group step produces good results. So, if we could automatically evaluate wether the results of the first step were good enough, we would not need to perform the other steps.

- Other options to group the references in the author grouping step. We may investigate unsupervised clustering techniques that produce pure clusters or how other attributes (work and publication venue title) may be used to produce pure clusters in the author group step.

- Other supervised techniques to be applied to the author assignment step. We may investigate how we can use/adapt other supervised clustering techniques to infer new authors (i.e., new classes) and reliable predictions to be used in the author assignment step.

- Options to adapt SAND to work in a incremental manner, i.e., to disambiguate only the references of the new citation records inserted into the digital library, avoiding the need for disambiguating all references of the digital library at once.

- How to adapt/generalize SAND to disambiguate other applications, e.g., ambiguous place names. If other applications have attributes highly discriminative, we may use such attributes to produce pure clusters in the first step and use the other attributes in the second and third steps.

- Options to use the feedback relevance indicated by the user to improve the disambiguation performance. We see two points where user feedback may be used: in the cluster selection and author assignment steps. In the cluster selection step, we may ask the user whether two clusters belong to the same author or not. In the author assignment step, we may ask the user for the correct authors of unreliable predictions.

Regarding SyGAR, the following items may be the subject of further research:

- Including a more sophisticated set of features to add new authors to the digital library. Our tool generates references of new authors inheriting a part of the interest area of an existing author. This strategy mimics the case of a new author who, starting its publication career, follows part of the interests of one who will be a frequent coauthor (e.g., advisor or colleague). It is interesting to investigate a manner of generating new authors without using a profile of an existing author.

- Other options to dynamically change the authors' publication profiles. A possible option to change the profiles of an author is to change the list of her coauthors

and consequently her interest area. We may investigate how an author starts to publish with a new coauthor or how the authors end to publish together.

- Functions to allow specifying the percentage of coauthors with similar names that publish with different authors with similar names and the percentage of publications to be generated that do not have coauthor names that are similar to the ones already inserted into the digital library.
- A manner of specifying the ambiguity of the load to be generated. If we can measure and define the degree of ambiguity of a load, we may investigate how the methods behave by increasing or decreasing the degree of ambiguity of a load.

Bibliography

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207--216, Washington, USA.

Artiles, J., Borthwick, A., Gonzalo, J., Sekine, S., and Amigó, E. (2010). Weps-3 evaluation campaign: Overview of the web people search clustering and attribute extraction tasks. In *CLEF 2010 LABs and Workshops, Notebook Papers*, Padua, Italy.

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

Bagga, A. and Baldwin, B. (1998). Algorithms for scoring coreference chains. In *Proceedings of the Seventh Message Understanding Conference (MUC7)*, pages 563--566.

Bekkerman, R. and McCallum, A. (2005). Disambiguating web appearances of people in a social network. In *Proceedings of the 14th International Conference on World Wide Web*, pages 463--470, Chiba, Japan.

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

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

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993--1022.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579--1619.

Bruno, N. and Chaudhuri, S. (2005). Flexible database generators. In *Proceedings of the International Conference on very large data bases*, pages 1097--1107, Trondheim, Norway. VLDB Endowment.

Carvalho, A. P., Ferreira, A. A., Laender, A. H. F., and Gonçalves, M. A. (2011). Incremental unsupervised name disambiguation in cleaned digital libraries. *Journal of Information and Data Management*, 2(3):289–304.

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

Christen, P. (2005). Probabilistic data generation for deduplication and data linkage. In *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, volume 3578 of *Lecture Notes in Computer Science*, pages 109–116, Brisbane, Australia. Springer.

Christen, P. (2008). Febrl -: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1065–1068, Las Vegas, Nevada, USA. ACM.

Christen, P. and Pudjijono, A. (2009). Accurate synthetic generation of realistic personal information. In *Proccedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 5476 of *Lecture Notes in Computer Science*, pages 507–514, Bangkok, Thailand. Springer.

Cohen, W. W., Ravikumar, P. D., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*, pages 73–78, Acapulco, Mexico.

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

Cota, R. G., Ferreira, A. A., Gonçalves, M. A., Laender, A. H. F., and Nascimento, C. (2010). An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. *Journal of the American Society for Information Science and Technology*, 61(9):1853--1870.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951--991.

Culotta, A., Kanani, P., Hall, R., Wick, M., and McCallum, A. (2007). Author disambiguation using error-driven machine learning with a ranking loss function. In *Proceedings of the International Workshop on Information Integration on the Web*, Vancouver, Canada.

Dempster, A., Laird, N., Rubin, D., et al. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1--38.

Diehl, C. P., Getoor, L., and Namata, G. (2006). Name reference resolution in organizational email archives. In *Proceedings of the SIAM International Conference on Data Mining*, pages 70--91, Bethesda, MD, USA.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226--231, Portland, Oregon.

Fan, X., Wang, J., Pu, X., Zhou, L., and Lv, B. (2011). On graph-based name disambiguation. *ACM Journal of Data and Information Quality*, 2:10:1--10:23.

Ferreira, A. A., Gonçalves, M. A., Almeida, J. M., Laender, A. H. F., and Veloso, A. (2009). SyGAR - A Synthetic Data Generator for Evaluating Name Disambiguation Methods. In *Proceedings of the 13th European Conference on Digital Libraries*, pages 437--441, Corfu, Greece.

Ferreira, A. A., Gonçalves, M. A., Almeida, J. M., Laender, A. H. F., and Veloso, A. (2012a). A tool for generating synthetic authorship records for evaluating author name disambiguation methods. *Information Sciences*, 206:42–62.

Ferreira, A. A., Gonçalves, M. A., and Laender, A. H. F. (2012b). A brief survey of automatic methods for author name disambiguation. *SIGMOD Record*, 41(2):15--26.

Ferreira, A. A., Silva, R., Gonçalves, M. A., Veloso, A., and Laender, A. H. F. (2012c). Active associative sampling for author name disambiguation. In *Proceedings of the 2012 ACM/IEEE Joint Conference on Digital Libraries*, pages 175--184, Washington, DC.

Ferreira, A. A., Veloso, A., Gonçalves, M. A., and Laender, A. H. F. (2010). Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 2010 ACM/IEEE Joint Conference on Digital Libraries*, pages 39–48, Gold Coast, Queensland, Australia.

Freund, Y. and Schapire, R. (1999). Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277--296.

Frey, B. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972--977.

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

Geisser, S. (1993). *Predictive inference: An introduction*. Chapman & Hall, New York.

Griffiths, T. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(1):5228--5235.

Han, H., Giles, C. L., Zha, H., Li, C., and Tsioutsiouliklis, K. (2004). Two supervised learning approaches for name disambiguation in author citations. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 296--305, Tuscon, USA.

Han, H., Xu, W., Zha, H., and Giles, C. L. (2005a). A hierarchical naive Bayes mixture model for name disambiguation in author citations. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 1065--1069, Santa Fe, New Mexico, USA.

Han, H., Zha, H., and Giles, C. L. (2005b). Name disambiguation in author citations using a k-way spectral clustering method. In *Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries*, pages 334--343, Denver, CO, USA.

Han, J. and Kamber, M. (2005). *Data mining: concepts and techniques*. Morgan Kaufmann, San Francisco, CA, USA.

Hoag, J. E. and Thompson, C. W. (2007). A parallel general-purpose synthetic data generator. *SIGMOD Record*, 36(1):19--24.

Huang, J., Ertekin, S., and Giles, C. L. (2006). Efficient name disambiguation for large-scale databases. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 536--544, Berlin, Germany.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264--323.

Kanani, P., McCallum, A., and Pal, C. (2007). Improving author coreference by resource-bounded information gathering from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 429–434, Hyderabad, India.

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

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90:733--795.

Laender, A. H. F., Gonçalves, M. A., Cota, R. G., Ferreira, A. A., Santos, R. L. T., and Silva, A. J. C. (2008). Keeping a digital library clean: new solutions to old problems. In *Proceedings of the ACM Symposium on Document Engineering*, pages 257–262.

Lagoze, C. and de Sompel, H. V. (2001). The open archives initiative: building a low-barrier interoperability framework. In *Proceedings of the 1st ACM/IEEE-CS Joint International Conference on Digital Libraries*, pages 54--62, Roanoke, Virginia, USA. ACM Press.

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

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

Lee, D., On, B.-W., Kang, J., and Park, S. (2005). Effective and scalable solutions for mixed and split citation problems in digital libraries. In *Proceedings of the 2nd International Workshop on Information Quality in Information Systems*, pages 69--76, Baltimore, Maryland.

Lee, M.-L., Ling, T. W., and Low, W. L. (2000). IntelliClean: a knowledge-based intelligent data cleaner. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 290--294.

Levin, F. H. and Heuser, C. A. (2010). Evaluating the use of social networks in author name disambiguation in digital libraries. *Journal of Information and Data Management*, 1(2):183--197.

Levin, M., Krawzyk, S., Bethard, S., and Jurafsky, D. (2012). Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, 63(5):1030--1047.

Li, H., Lee, W.-C., Sivasubramaniam, A., and Giles, C. L. (2007). SearchGen: A Synthetic Workload Generator for Scientific Literature Digital Libraries and Search Engines. In *Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries*, pages 137--146, Vancouver, BC, Canada.

Liming, L. and Lihua, L. (2005). Scientific publication activities of 32 countries. *Scientometrics*, 26(2):263--273.

Malin, B. (2005). Unsupervised name disambiguation via social network similarity. In *Proceedings of the Workshop on Link Analysis, Counterterrorism, and Security, at the SIAM International Conference on Data Mining*, pages 93--102, Newport Beach, CA.

McKay, D., Sanchez, S., and Parker, R. (2010). What's my name again?: sociotechnical considerations for author name management in research databases. In *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*, pages 240--247, Brisbane, Australia.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York, NY, USA.

Oliveira, J. W. A. (2005). A strategy for removing ambiguity in the identification of the authorship of digital objects. Master's thesis, UFMG, Belo Horizonte, Brazil. (in Portuguese).

On, B.-W., Elmacioglu, E., Lee, D., Kang, J., and Pei, J. (2006). Improving grouped-entity resolution using quasi-cliques. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 1008--015, Hong Kong, China. IEEE Computer Society.

On, B.-W. and Lee, D. (2007). Scalable name disambiguation using multi-level graph partition. In *Proceedings of the 7th SIAM International Conference on Data Mining*, pages 575--580, Minneapolis, Minnesota, USA.

On, B.-W., Lee, D., Kang, J., and Mitra, P. (2005). Comparative study of name disambiguation problem using a scalable blocking-based framework. In *Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries*, pages 344--353, Denver, CO, USA.

Pereira, D. A., Ribeiro-Neto, B. A., Ziviani, N., Laender, A. H. F., Gonçalves, M. A., and Ferreira, A. A. (2009). Using web information for author name disambiguation. In *Proceedings of the 2009 ACM/IEEE Joint Conference on Digital Libraries*, pages 49–58, Austin, TX, USA.

Rijsbergen, C. J. V. (1979). *Information Retrieval, 2nd edition*. Butterworths, London.

Rosen-Zvi, M., Griffiths, T. L., Steyvers, M., and Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the Conference in Uncertainty in Artificial Intelligence*, pages 487–494, Banff, Canada.

Salton, G. M., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613--620.

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

Shu, L., Long, B., and Meng, W. (2009). A latent topic model for complete entity resolution. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 880--891, Shanghai, China. IEEE Computer Society.

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

Song, Y., Huang, J., Councill, I. G., Li, J., and Giles, C. L. (2007). Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries*, pages 342--351, Vancouver, BC, Canada.

Tang, J., Fong, A. C. M., Wang, B., and Zhang, J. (2012). A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):975–987.

Torvik , V. I., Weeber, M., Swanson, D. R., and Smalheiser, N. R. (2005). 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.

Torvik, V. I. and Smalheiser, N. R. (2009). Author name disambiguation in medline. *ACM Transactions on Knowledge Discovery from Data*, 3(3):1--29.

Treeratpituk, P. and Giles, C. L. (2009). Disambiguating authors in academic publications using random forests. In *Proceedings of the 2009 ACM/IEEE Joint Conference on Digital Libraries*, pages 39--48, Austin, TX, USA.

Veloso, A., Ferreira, A. A., Gonçalves, M. A., Laender, A. H., and Meira Jr., W. (2012). Cost-effective on-demand associative author name disambiguation. *Information Processing & Management*, 48(4):680 – 697.

Veloso, A., Meira Jr., W., Cristo, M., Gonçalves, M., and Zaki, M. (2006a). Multi-evidence, multi-criteria, lazy associative document classification. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management*, pages 218--227, Arlington, USA.

Veloso, A., Meira Jr., W., and Zaki, M. J. (2006b). Lazy associative classification. In *Proceedings of the International Conference on Data Mining*, pages 645--654, Washington, DC, USA.

Vu, Q. M., Masada, T., Takasu, A., and Adachi, J. (2007). Using a knowledge base to disambiguate personal name in web search results. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 839--843, Seoul, Korea.

Yang, K.-H., Peng, H.-T., Jiang, J.-Y., Lee, H.-M., and Ho, J.-M. (2008). Author name disambiguation for citations using topic and web correlation. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries*, pages 185--196, Aarhus, Denmark. Springer-Verlag.

Yoshida, M., Ikeda, M., Ono, S., Sato, I., and Nakagawa, H. (2010). Person name disambiguation by bootstrapping. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 10–17, Geneva, Switzerland.

Zha, H., He, X., Ding, C. H. Q., Gu, M., and Simon, H. D. (2001). Spectral relaxation for K-means clustering. In *Neural Information Processing Systems*, pages 1057–1064. MIT Press.