The following page is the Signature Page. The Signature Page needs to be given to the Grad School, but it should not be bound with the thesis.

Don't bind this page either!

We approve the thesis of Byung-Won On.

Date of Signature

Dongwon Lee
Assistant Professor of Information Sciences and Technology
Thesis Advisor, Chair of Committee

C. Lee Giles
David Reese Professor of Information Sciences and Technology

Sandeep Purao
Associate Professor of Information Sciences and Technology

Wang-Chien Lee
Associate Professor of Computer Science and Engineering

Piotr Berman
Associate Professor of Computer Science and Engineering

Reka Albert
Assistant Professor of Physics

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

The Pennsylvania State University
The Graduate School

DATA CLEANING TECHNIQUES BY MEANS OF ENTITY RESOLUTION

A Thesis in
Computer Science and Engineering
by
Byung-Won On

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2007

The thesis of Byung-Won On was reviewed and approved* by the following:

Dongwon Lee
Assistant Professor of Information Sciences and Technology
Thesis Advisor, Chair of Committee

C. Lee Giles
David Reese Professor of Information Sciences and Technology

Sandeep Purao
Associate Professor of Information Sciences and Technology

Wang-Chien Lee
Associate Professor of Computer Science and Engineering

Piotr Berman
Associate Professor of Computer Science and Engineering

Reka Albert
Assistant Professor of Physics

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

Real data are "dirty." Despite active research on integrity constraints enforcement and data cleaning, real data in real database applications are still dirty. To make matters worse, both diverse formats/usages of modern data and demands for large-scale data handling make this problem even harder. In particular, to surmount the challenges for which conventional solutions against this problem no longer work, we focus on one type of problems known as the Entity Resolution (ER) – the process of identifying and merging duplicate entities determined to represent the same real-world object. Despite the fact that the problem has been studied extensively, it is still not trivial to de-duplicate complex entities among a large number of candidates.

In this thesis, we have studied three specialized types of ER problems: (1) the Split Entity Resolution (SER) problem, in which instances of the same entity type mistakenly appear under different name variants; (2) the Mixed Entity Resolution (MER) problem, in which instances of different entities appear together for their homonymous names; and (3) the Grouped Entity Resolution (GER) problem, in which instances of entities do not carry any name or description by which ER techniques can be utilized, and thus the contents of entities are exploited as a group of elements. For each type of problems, we have developed a novel scalable solution. Especially, for the GER problem, we have developed two graph theoretic algorithms - one based on Quasi-Clique and the other based on Bipartite Matching, and experimentally validate the superiority of the proposed solutions.

Table of Contents

List of Figures

List of Tables

List of Symbols

x, y	co-author names, p. 28
T_x	all tokens of the co-author x, p. 28
C_x	all characters of x, p. 28
$CC_{x,y}$	all characters in x common with y, p. 28
$X_{x,y}$	# of transpositions of char. in x relative to y, p. 28
AR	Average Recall, p. 53
ARP	Average Ranked Precision, p. 53
ME	MongeElkan string distance metric, p. 55
JR	Jaro string distance metric, p. 55
JC	Jccard string distance metric, p. 55
TI	TF/IDF Cosine similarity, p. 55
JS	Jensen-Shannon string distance metric, p. 55
FS	Fellegi-Sunter string distance metric, p. 55
ST	SoftTFIDF string distance metric, p. 55
CS	Cosine similarity, p. 55
IC	IntelliClean, p. 56

JC+QC	Jaccard + Quasi-Clique, p. 56
TI+QC	TF/IDF Cosine similarity + Quasi-Clique, p. 56
IC+QC	IntelliClean + Quasi-Clique, p. 56
D	relation of multi-attribute records, p. 61
g_1, g_2, ...	groups of records in D, p. 61
r_1, r_2, ...	records in D, p. 61
$sim(r_i, r_j)$	arbitrary record-level similarity function, p. 61
θ	group-level similarity threshold, p. 61
ρ	record-level similarity threshold, p. 61
M	maximum weight bipartite matching, p. 61
BM	bipartite matching based group linkage, p. 61
UB, LB	upper and lower bound of BM, p. 61
MAX	$max()$ based heuristic group linkage, p. 61
JA	Jaccard based group linkage measure, p. 77
BM	$BM_{sim,\rho}$ group linkage measure, p. 77
UB	$UB_{sim,\rho}$ group linkage measure, p. 77
MAX	$MAX_{sim,\rho}$ group linkage measure, p. 77
k	# of clusters, p. 92
l	# of tokens, p. 92
m	# of unique tokens (i.e., m-dimensional vectors), p. 92
n	# of documents (i.e., entities), p. 92
c	Constant, p. 92

Acknowledgments

Writing this dissertation marks the happy conclusion of the journey that I started many years ago. Throughout the journey, I greatly and sincerely benefited from the support and companionship of many people.

First and foremost, I would like to thank my parents, Namsun On and Sunja Hwang, and my sisters and my brother-in-laws, Eunyoung On and Jaehyun Hwang, and Sooyoung On and Jaemin Han, from the bottom of my heart, for their prayers and confidence in me. I am really proud of them all.

In addition, I would like to express my gratitude to Dr. Dongwon Lee, Assistant Professor of the College of Information Sciences and Technology, the Pennsylvania State University for his support and encouragement throughout my Ph.D study. He is an enthusiastic researcher who is always looking for novel perspectives and motivating his students toward innovations. I have tried to learn his insightful perspectives for the last five years since I started working with him. He also provided the unmeasurable guidance and feedback on the research problems and issues studied in this dissertation. I would also like to thank the other committee members, Dr. C. Lee Giles, Dr. Sandeep Purao, Dr. Piotr Berman, Dr. Wang-Chien Lee, Dr. Reka, and Dr. Raj Acharya, for their guidance and suggestions during my research.

Especially I would like to thank Vicki Keller, the graduate secretary in the Department of Computer Science and Engineering. During my Ph.D. program, she had really helped and advised me. Without her generous encouragement, I could not imagine that I could finish my work.

Finally, I would like to acknowledge the partial support provided by Microsoft SciData Award (2005) and IBM Eclipse Innovation Award (2006). The findings and reporting are my own conclusion and does not reflect the positions of the above funding agencies.

Chapter 1

Introduction

Large-scale data repositories often suffer from duplicate entities whose representations are different, but yet refer to the same real world object. For instance, in digital libraries (e.g., ACM [24], DBLP [7], CiteSeer [49], arXiv e-Print [2], etc.), there may be various name variants of an author due to errors introduced in data entry or errors from imperfect data collection softwares. Let us denote that, among the duplicate entities, the single authoritative one as *canonical entity* while the rest as *variant entities* or *variants* in short. Since the existence of variant entities degrades the quality of the collection severely, it is important to deduplicate them. Such a problem is, in general, known as the **Entity Resolution (ER)** problem. The ER problem frequently occurs in many applications and is exacerbated especially when data are integrated from heterogeneous sources. Note that the ER problem cannot be completely avoided since not all entities in data collections carry an ID system such as digital object ID (DOI) [54].

The ER problem can be categorized into two distinct cases. In the *split entity* case, an entity is recorded under various variants. Thus the goal of this case is to detect all variant entities, and consolidate them. On the other hand, in the *mixed entity* case, the contents of different entities are mixed in the same pool. Therefore, the goal is to group different contents into different clusters. For instance, in DBLP [7], there are at least four and at most eleven "Wei Wang"s. Since all share the same name spelling, all of their citations are mixed together, causing a confusion. Furthermore, in the *grouped entity* case, each entity has "a list of tuples" associated with it. Examples include an author entity with a list of

citations, a singer entity with song list, or an intermediate result by GROUP BY SQL query. Thus, the goal is to resolve entities that contain "a group of elements" in them.

1.1 Motivation

In order to demonstrate the need for a solution to the ER problem, let us present the following real cases drawn from a wide variety of applications.

1. Figure 1.1 is a screen shot of the ACM Portal, which contains the list of author names who have ever published an article in ACM-affiliated conferences or journals. In particular, Figure 1.1 shows names whose last names are either "Ullman" or "Ullmann". Note that the name of the renowned computer scientist, "Jeffrey D. Ullman" at Stanford university, appears as several variants, *incorrectly*. For instance, "J. D. Ullman" in the list is in fact the same person as "Jeffrey D. Ullman", however they are treated as different scholars.

2. Recently data integration is a promising research topic in the database community. To integrate data from multiple independent, heterogeneous data sources, we need to resolve main challenging conflict. For instance, Figure 1.2 shows the screen-shot of different author name formats between DBLP and ACM in Microsoft SQL Server. For example, "Clement T. Yu" (name1 field) in DBLP and "C. T. Yu" (name2 field) in ACM by name abbreviation. They have different name conventions.

3. In the real world, duplicates refering to the same entity occur due to (1) typographical errors; (2) abbreviated, imcomplete, or missing information – e.g., "New Jersey" and "NJ", "Peter Norvig" and "P. Norvig", "Very Large Data Bases 2007" and "VLDB07", "Western Digital 120GB 7200RPM" and "WD 120GB 7200RPM"; (3) different formatting conventions – e.g., "Street" and "St.", "Mary Irwin" and "Irwin, Mary"; (4) homonym – e.g., "cost" and "price"; (5) noise information – e.g., "1995" and "<year>1995</year>", "Artificial Intelligence – a Modern Approach" and "Artificial Intelligence: A

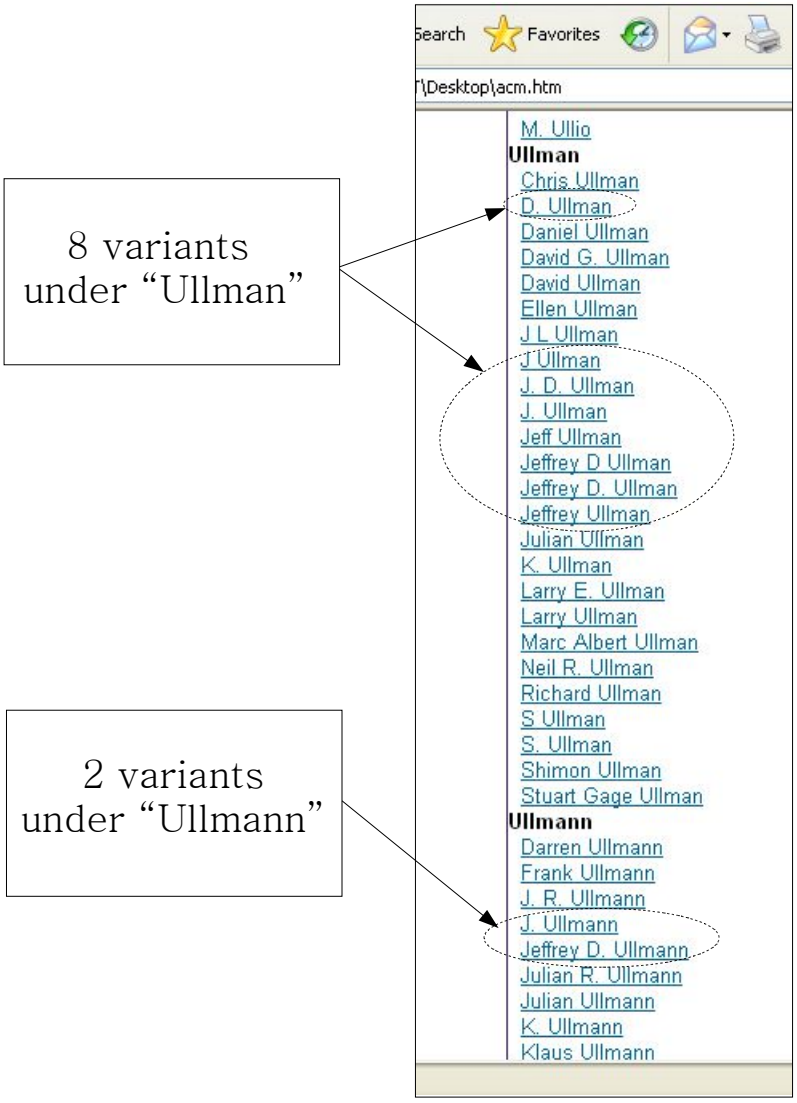


Figure 1.1: Screen-shot of author index for "Ull*" in the ACM Portal. Note that the citations of "Jeffrey D. Ullman" appear as eight variants under "Ullman" and two variants under "Ullmann."

Modern Approach", "..., WWW" and "..., WWW 2006 (Best Paper Award)"; and (6) combinations thereof. Figure 1.3 illustrates the multiple variants of author names in ACM because of name abbreviations and typos. Similarly, Figure 1.4 shows the real case in which an author name "Alon Y. Levy" was replaced by a new name "Alon Y. Halevy" in DBLP.

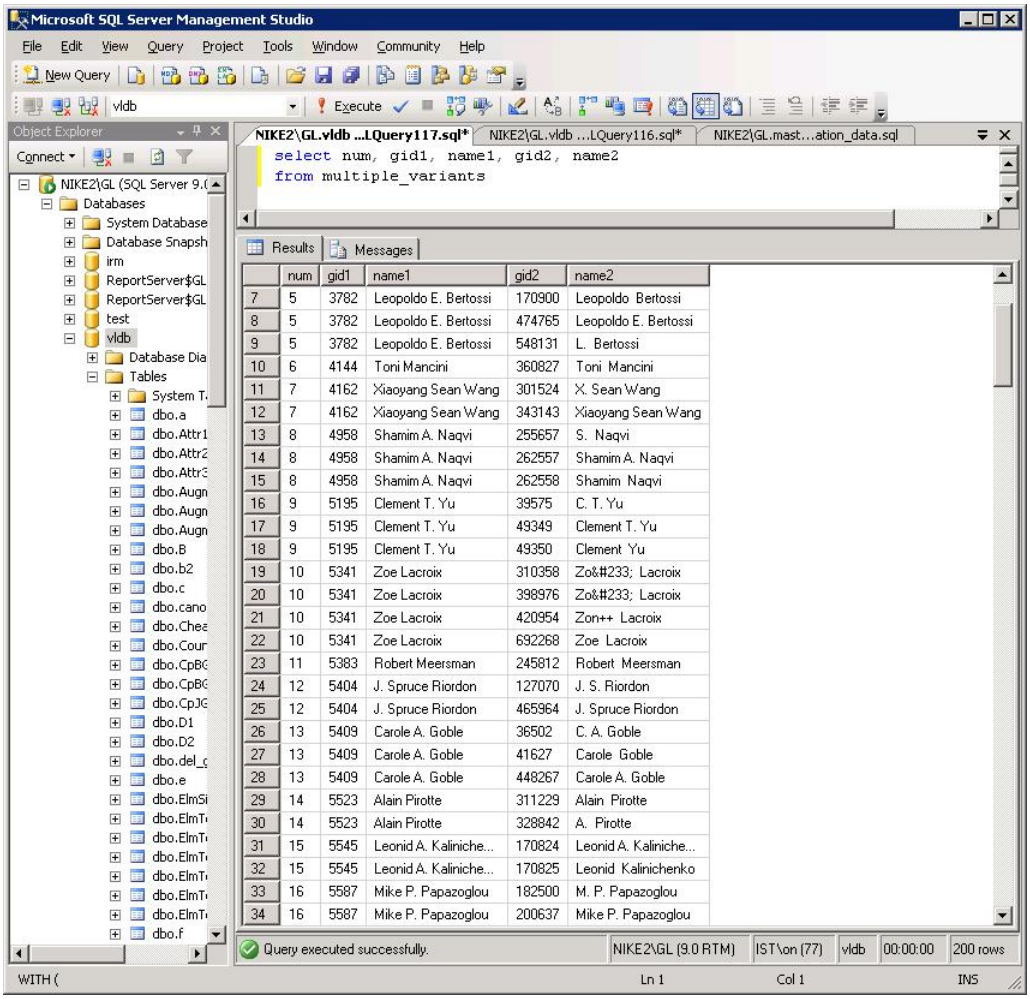
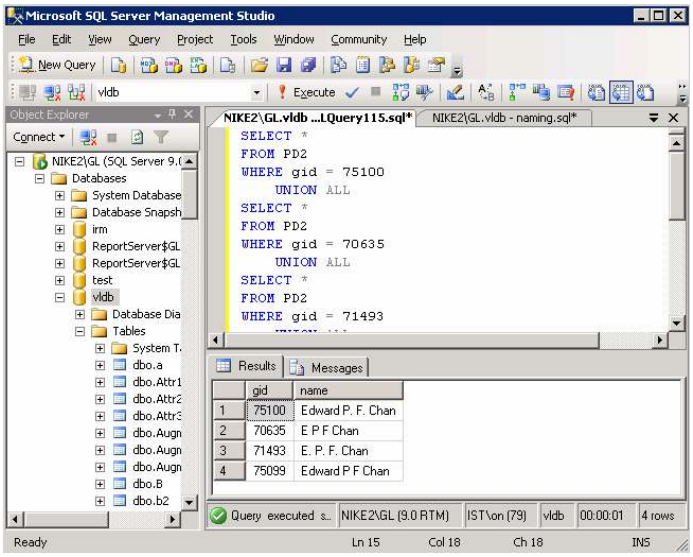
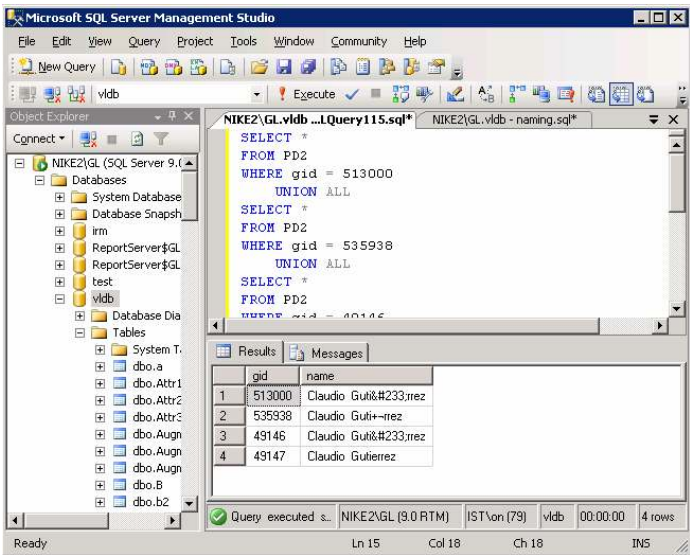


Figure 1.2: Screen-shot of different author name formats between DBLP [7] and ACM [24] in MS SQL Server (e.g., "Clement T. Yu" (name1 field) in DBLP and "C. T. Yu" (name2 field) in ACM).

4. The Split Entity Resolution (SER) problem is ubiquitous in many domains. This problem also occurs in popular search engines. Figure 1.5 is drawn from CiteSeer, where we tried to locate all citations about a book, "Artificial Intelligence: A Modern Approach", by "S. Russell" and "P. Norvig". As illustrated in Figure 1.5, at the time of the search, CiteSeer returns 23 different formats of citations of the same book, *incorrectly* thinking that they are all different. Part of the problem is caused by the ambiguous names of the authors. Similarly, Google Scholar returns redundant citations referring to the same book.



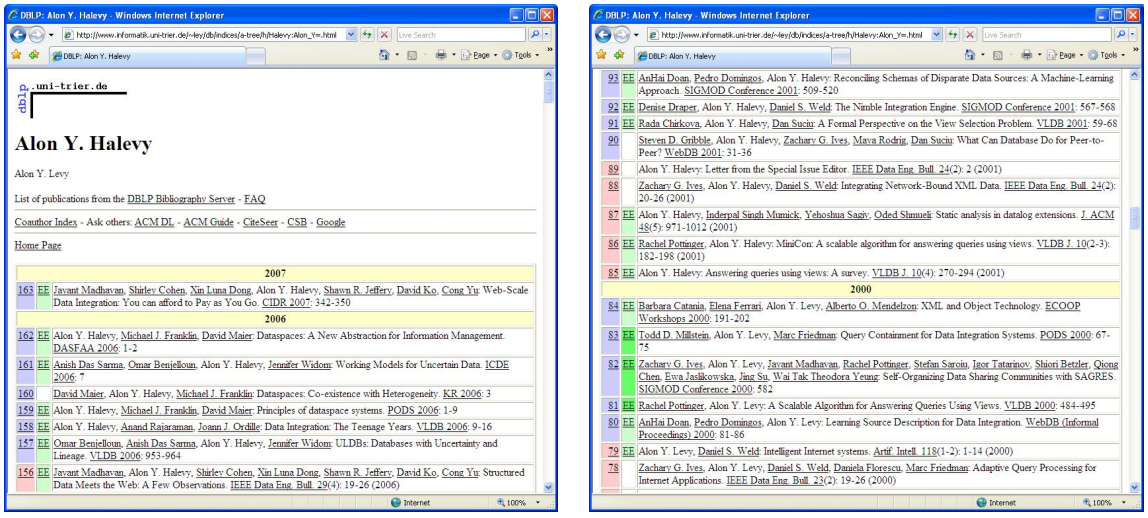
(a) Multiple variants of "Edward Chan" by name abbreviations.



(b) Multiple variants of "Claudio Gutierrez" by typos.

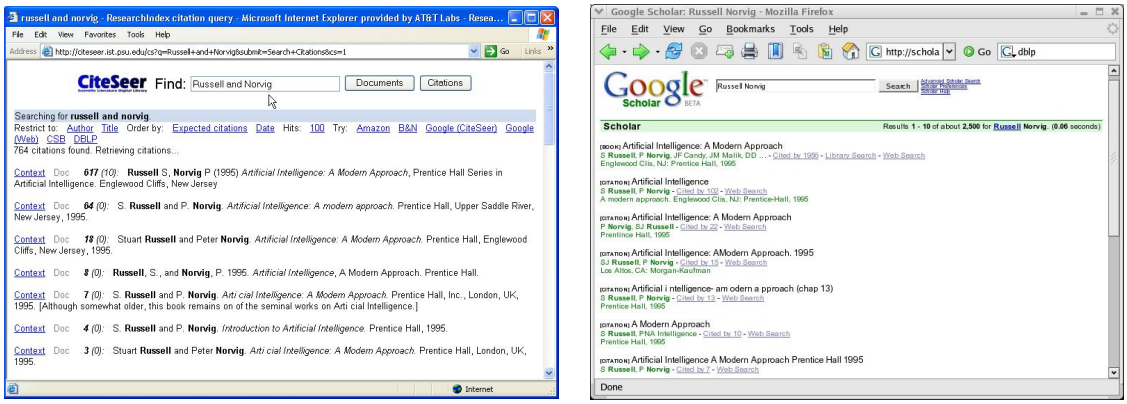
Figure 1.3: Multiple name variants in ACM [24].

5. Duplicate records exist in the bio-medical informatics domain. As illustrated in Table 1.1, one sequence is from UniProt [67] and the other is from Bond [9]. Both databases are constructed to provide comprehensive information for biological sequence. However, although two sequences from UniProt and Bond are identical, they are slightly different as shown in the example.



(a) Alon Y. **Halevy** (2001 - present) (b) Alon Y. **Levy** (by 2000)

Figure 1.4: An example of current name "Alon Y. Halevy" and old name "Alon Y. Levy".



(a) CiteSeer (b) Google Scholar

Figure 1.5: Screen-shot of citation search for "Russell and Norvig" in CiteSeer (a) and Google Scholar (b).

6. Grouped-Entity Resolution is a specialized problem of the Entity Resolution problem. Many entities contain "a group of elements" in them. We refer to such an entity as the Grouped-Entity and the ER problem on grouped-entity as Grouped-Entity Resolution (GER) problem. Figure 1.6 illustrates a typical example of the GER problem from DBLP. Although "Yinfeng Xu" and "Yin-Feng Xu" are the same scholar in Computer Science and Engineering, such two names are split due to an error of name contraction. Here, an en-

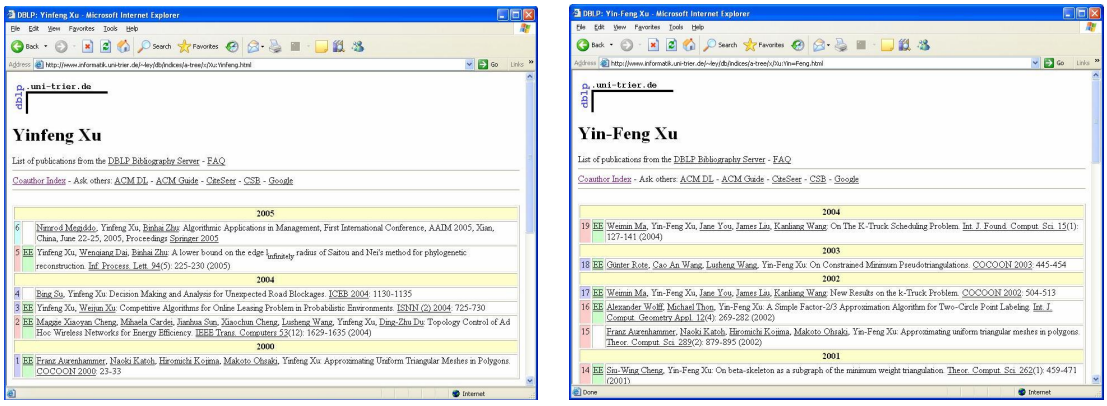
Table 1.1: The redundant sequences in bio medical informatics.

	UniProt	Bond
Accession	P38910	6324594
Title	10kDa heat shock protein, mitochondrial	Hsp10p [Saccharomyces cerevisiae]
Synonyms	HSP10, 10kDa chaperonin	Hsp10p, Mitochondrial matrix co-chaperonin that inhibits the ATPase activity of Hsp60p, a mitochondrial chaperonin; involved in protein folding and sorting in the mitochondria; 10kD heat shock protein with similarity to E. coli groES, Mitochondrial matrix co-chaperonin that inhibits the ATPase activity of Hsp60p, a mitochondrial chaperonin; involved in protein folding and sorting in the mitochondria; 10 kD heat shock protein with similarity to E. coli groES; Hsp10p [Saccharomyces cerevisiae], HSP10, CPN10
Source	Saccharomyces cerevisiae (Baker's yeast)	Baker's yeast
Sequence	MSTLLKSAKSIVPLMDRVLV QRIKAQAKTASGLYLPEKNV EKLNQAEVVAVGPGFTDANG NKVVPQVKVGDQVLIPQFGG STIKLGNDDEVILFRDAEIL AKIAKD	MSTLLKSAKSIVPLMDRVLV QRIKAQAKTASGLYLPEKNV EKLNQAEVVAVGPGFTDANG NKVVPQVKVGDQVLIPQFGG STIKLGNDDEVILFRDAEIL AKIAKD

tity "Yinfeng Xu" has 6 citations and the other entity "Yin-Feng Xu" has 19 citations. In this case, each entity (i.e., "Yinfeng Xu" or "Yin-Feng Xu") includes a group of citations in it.

7. The next problematic example is an inverse case of both split entity and group-entity examples. It is drawn from DBLP, where users can browse a collection of articles grouped by author's full name (i.e., author's full name acts as a primary key). In particular, Figure 1.7 is a screen-shot of a collection of articles by "Wei Wang". However, there are at least four (possibly up to eleven) active computer scientists with the same name spelling of "Wei Wang." Not surprisingly, their citations are all mixed here. Any bibliometric analysis using this data would be, needless to say, faulty. When we manually checked this mixed entity cases in DBLP, we gathered many real cases [1]. "H. Cai", "Wei Cai", "John M. Carroll", "Li Chen", "Yu Chen",

[1]We directly requested authors to verify if their citations are mixed with different scholars with the same name spellings.



(a) **Yin Feng** Xu (b) **Yin-Feng** Xu

Figure 1.6: An example of the Grouped-Entity problem.

"Hui Han", "Youngjae Kim", "Dongwon Lee", "Chen Li", "Jia Li", "Jian Li", "Lin Li", "Peng Liu", "Wei Liu", "Zhenyu Liu", "Jiebo Lou", "Murali Mani", "Prasenjit Mitra", "Sanghyun Park", "Hui Song", "James Ze Wang", "Wei Wang", "Yuan Xie", "Wei Xu", and "Bo Luo" are such authors. This indicates that the mixed entity problem frequently occurs in Digital Libraries (DLs).

8. The mixed entity problem does not occur only in digital libraries, such as DBLP, ACM, BioMed, and so forth. Figure 1.8 illustrates a real case where entities are mixed due to their name homonyms. There are at least 41 movies with the title "Love" in Internet Movie Data Base (IMDB) [38].

9. The mixed entity problem is one of the main issues in Information Retrieval. According to the U.S. Census Bureau, only 90,000 different names are shared by 100 million people and about 30% of search engine queries include person names. This indicates that the search results are a mixture of web pages about different people with the same name spellings. Thus, ideal search engines should cluster web pages by different people sharing the same name. For example, Bekkerman et al. [4] introduced interesting examples, in which "Tom Mitchell" is issued as a query to Google [27], 92 web pages are retrieved. Among these 92 documents, there are 37 namesakes to "Tom Mitchell". For example, "Tom Mitchell" appears as musicians, executive managers, astrologist, a hacker, and a rabbi. A set of 32 entities are mixed since they all have

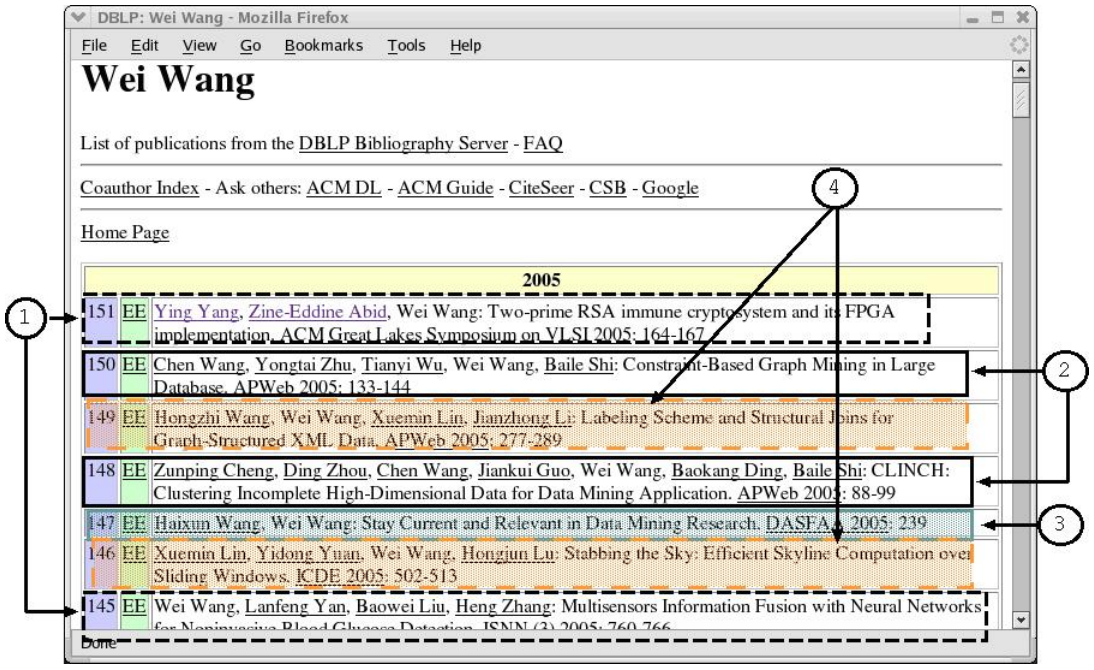


Figure 1.7: Screen-shot of a collection of citations under author "Wei Wang" in DBLP. Note that there are at least four distinct computer scientists with the same name "Wei Wang."

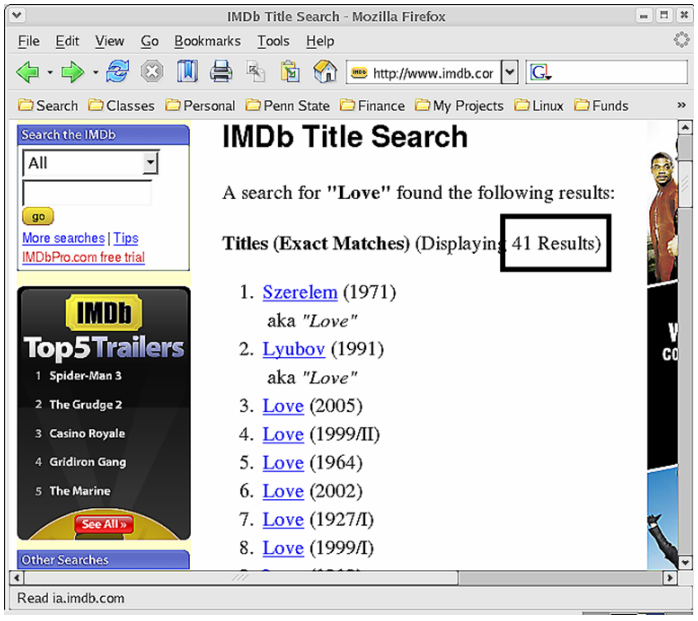


Figure 1.8: Screen-shot of mixed entities in IMDB.

Figure 1.9: Three bus images and one historical remains image [48, 69].

the same name description of "Tom Mitchell".

10. In the area of image information retrieval, effective solutions for the mixed entity resolution problem are also required. In order to locate and download an online copy of a particular image, one typically uses image search engines with keywords. As illustrated in Figure 1.9, suppose that the local database of an image search engine contains four images, three red bus images and one historical remains image. When we want to find bus images, entity resolution methods can cluster the entire set of images to similar categories and return the correct category to us.

These real examples indicate that the Split Entity and Mixed Entity Resolution problems frequently occurs in many applications, and entity resolution solutions should be developed to handle these problems. In our context, we focus on "scientific literature Digital Libraries (DLs)".

Table 1.2: The data size of modern digital libraries.

DL	Domain	Size (in M)
ISI/SCI	General Sciences	25
CAS	Chemistry	23
Medline/PubMed	Life Science	12
CiteSeer	General Sciences/Engineering	10
arXiv	Physics/Math	0.3
SPIRED HEP	Physics	0.5
DBLP	CompSci	0.8
CSB	CompSci	1.4

1.2 Contributions

In general, these ER problems prevail in the current bibliographic DLs (e.g., DBLP [7], CiteSeer [49], or e-Print arXiv [2], etc.). Note that these problems have been exacerbated as the number and volume of DLs increase, posing significant challenges to the management of large-scale DLs, as shown in Table 1.2. One of the key challenges is to make the ER algorithms *scalable* and *resilient* in order to cope with the rapid growth of the DLs while not degrading their accuracy. The traditional approaches that rely on the syntactic similarity of name variants (e.g., measuring string edit-distance [14] between two name entities) will likely fail to scale because, as the size of DLs increases, more numbers of common or similar names will appear in DLs, making it increasingly difficult to distinguish them using only their names.

To investigate these problems, in this thesis, we introduce a two-step framework, where many alternative methods can be used in each step in order to optimize the performance. We identify the contributions of our work as follows:

1. If all entity names (i.e., author names in our context) have to be compared against each other in order to find similar names, the resulting algorithm will be quadratic to the total number of entity names in the input. Clearly, this algorithm will not be scalable for large DLs. In order to address this problem, we advocate a scalable two-step SER framework. In the first step, the algorithm partitions all entity-name strings into a set of disjoint blocks. In the second step, the algorithm visits each block and compare all possible pairs of names within the block in order to find the name variants. This

blocking-based pruning method has been introduced and successfully applied in other problem contexts such as record linkage [23] or identity uncertainty problems [55].

2. In our Split Entity Resolution (SER) and Mixed Entity Resolution (MER) framework, we focus on a set of "unsupervised" distance measures other than supervised algorithms. Using citations from large real-world DLs, we show that, some of the unsupervised algorithms (e.g., cosine[2] or TF/IDF Cosine similarity) show comparable or even superior accuracy to supervised ones (e.g., the Naive Bayes model [31] or the Support Vector Machines [31]). Since unsupervised algorithms in general do not require large training sets, they are useful for resolving two entities. In addition, we exploit information associated with entities (e.g., co-authors) in order to further improve the algorithm's accuracy. This approach is based on the hypothesis that using associated information more than name entities themselves would be beneficial in resolving name variants.

3. We formulate the GER problem as a specialized form of the ER problem. Since the grouped entities in the GER problem contain a wealth of information (i.e., a group of elements), its exploitation can result in better outcome. More specifically, we introduce how to capture "contextual information" hidden in a group of elements in grouped entities. In particular, we propose to use the technique of *superimposition* to mine hidden relationships into graphs. Furthermore, to capture the "contextual distance" between grouped entities, we exploit the notion of *Quasi-Clique* [56], a measure to see how strong inter-relationships between two graphs are, and propose a simple yet effective two-step GER algorithm, distQC.

4. Often entities are represented as groups of relational records, rather than individual relational records, e.g., households in a census survey consist of a group of persons. We refer to the problem of determining if two entities, represented as groups, are approximately the same as *group linkage*. Intuitively, two groups can be linked to each other if (i) there is high enough

[2]Cosine is the abbreviated term of Cosine similarity in this thesis.

similarity between "matching" pairs of individual records that constitute the two groups, and (ii) there is a large fraction of such matching record pairs. In this problem, we formalize this intuition and propose a group linkage measure based on bipartite graph matching [15]. Given a data set consisting of a large number of groups, efficiently finding groups with a high group linkage similarity to an input query group requires quickly eliminating the many groups that are unlikely to be desired matches. To enable this task, we present simpler group similarity measures that can be used either during fast pre-processing steps or as approximations to our proposed group linkage measure. These measures can be easily instantiated using SQL, permitting our techniques to be implemented inside the database system itself.

5. For solutions to the MER problem, many clustering approaches have been proposed [31, 51, 32, 3]. However, since such methods are not scalable, we propose a scalable citation labeling algorithm using the sampling based approximated join algorithm to quickly determine a small number of candidates from an entire set of entities. As an alternative method, we propose a scalable name disambiguation algorithm using multi-level graph partition [3]. According to Han et al. [32], k-way spectral clustering algorithm is the most effective method. However, as the size of a graph is significantly huge, it takes a large amount of time. To speed up such a graph partitioning algorithm but yet optimize clusters, we apply the multi-level graph partitioning algorithm to the MER problem.

In essence, the ER problems cannot be completely avoided unless each entity carries a universal ID. Note that these problems would still occur even if digital object identifier (DOI) system is fully adopted, since it usually does not govern the identity of an entity or its name. In this thesis, we plan to investigate effective solutions for these problems.

The rest of this thesis is organized as follows. In Chapter 2, we review previous work related to our research. In Chapter 3, we discuss the Split Entity Resolution problem. In Chapter 4, we discuss the Grouped-Entity Resolution problem. Then,

[3]The disadvantage of classification methods is that training data sets are required. To surmount such limitation, we focus on designing and developing novel clustering methods in the MER problem.

the discussion to the Mixed Entity Resolution problem is given in Chapter 5. Finally, we conclude this work in Chapter 6.

Chapter 2

Background Literature Survey

In this chapter we present an overview of previous research concerning the current research topic. The general ER problem has known as various names – record linkage (e.g., [23, 8]), citation matching (e.g., [52]), identity uncertainty (e.g., [55]), merge-purge (e.g., [34]), object matching (e.g., [11]), duplicate detection (e.g., [60, 1]), approximate string join (e.g., [29]) etc. Before we can process citations, we assume that field segmentation and identification has been completed using some methods like one in [10]. Blocking was first proposed by Kelley et al. [43] in the record linkage literature. Our blocking scheme is also similar in flavor to the two-step citation matching schemes proposed in [37, 52] where initial rough but fast clustering (or "Canopy") is followed by more exhaustive citation matching step. String similarity measures used in our work were proposed by Jaro [39] and Winkler [71]. Bilenko et al. have studied name matching for information integration [8] using string-based and token-based methods. Cohen et al. have also compared the efficacy of string-distance metrics, like JaroWinkler, for the name matching task [14, 63, 22]. In DLs, this problem is called citation matching. In the citation matching domain, [44] experimented with various distance-based algorithms with a conclusion that word based matching performs well.

2.1 The Split Entity Resolution (SER) Problem

ALIAS system in [60] proposes a framework to detect duplicate entities such as citations or addresses, but its focus is on the learning aspect. Unlike the exist-

ing entity matching solutions exploiting "syntactic similarities" (e.g., string distance metrics), *Constraint-based Entity Matching (CEM)* [61] examines "semantic constraints" in both scalable and unsupervised way. They use two popular data mining techniques, the *Expectation-Maximization (EM)* algorithm, estimating the parameters of the generative model on how data sets satisfying constraints are genrated, and *relaxation labeling*, exploiting the constraints. This framework differs from our approach because of the motivation of domain constraints and user interaction. [5] present a generic approach, named as *Swoosh* algorithms. In the algorithms, they consider the functions for matching and merging records as black boxes for a generic, extensible solution. The objective of these *Swoosh* algorithms is to minimize time complexity of black boxes, compared to naive algorithm. The recent work by [21] proposes an iterative ER solution for complex personal information management. Their work reports good performance for its unique framework where different ER results mutually reinforce each other (e.g., the resolved co-author names are used in resolving venue names). Although not directly comparable to our work, it would be interesting to see how the hybrid approach works. Another stream of works that are relevant to our work is name/entity disambiguation and authority controls in NLP community. For instance, works done in [70] aim at detecting name variants automatically using data mining or heuristics techniques, but do not consider the issue of scalability nor in the context of digital libraries. Similarly, [18] introduces a method to find matching variants of named entity in a given text such as project name (e.g., DBLP vs. Data Base and Logic Programming). [66] discusses an effort to standardize author names using a unique number, called INSAN, and [17] is a recent implementation for name authority control, called HoPEc. On the contrary, we focus more on two specific problems relevant to citations of digital libraries. In [35], we investigated issues related to system support for both problems.

2.2 The Grouped-Entity Resolution (GER) Problem

When entities are "grouped entities," existing ER methods do not distinguish them, while our **distQC** tries to exploit them using *Quasi-Clique*. Our method can be used together with any of these ER methods as the first step. The recent trend in the ER problem shows similar direction to ours (e.g., [13, 8, 6, 60, 51, 46]) – Although each work calls its proposal under different titles, by and large, most are trying to "exploit additional information beyond string comparison." A more extensive and systematic study is needed to investigate the usefulness and limitations of the context in a multitude of the ER problem. Especially, to capture "contexts," graph-based partitioning techniques have been exploited. Recent work by Kalashnikov et. al [40] presents a relationship-based data cleaning (RelDC) which exploits context information for entity resolution, sharing similar idea to ours. RelDC constructs a graph of entities connected through relationships and compares the connection strengths across the entities on the graph to determine correspondences. The main difference is the notion of data structure used (i.e., we used Quasi-Clique along with superimpostion to derive distances between graphs). Malin [51] presented a less strict similarity requirement by using random walks between duplicate entities on a global social network constructed from all sources, or a community similarity. Bhattacharya [6] make use of attribute similarity measures, but in addition, it takes into account the similarity of linked objects through the "iterative" deduplication process. However, their approach has the downside that the process is more expensive computationally.

2.3 The Mixed Entity Resolution (MER) Problem

There have been works focusing on the case of *mixed entities* such as [4, 45, 31] (i.e., how to cluster mixed citations into groups of distinct authors despite confusing author names). Han et al. [31] proposed two supervised learning-based approaches. Their algorithm solves the so called the *citation labeling* problem

– given a citation, cross out an author name, and using the remaining citation information, recover the author name via two supervised learning methods: (1) the Naive Bayes model; (2) the Support Vector Machines. Furthermore, Han et al. [32] presented an unsupervised learning approach using K-way spectral clustering that groups different citations into different clusters, using three types of citation attributes – co-author names, paper titles, and publication venue titles. In addition, Malin [51] utilizes hierarchical clustering, relying upon exact name similarity. However, all the approaches are not scalable to handle large-scale entities. As a scalable solution, Lee et al. [45] proposed a scalable citation labeling algorithm recently. In their algorithm, authors use the sampling-based technique to quickly determine a small number of candidates from the entire authors in a digital library. The idea of the citation labeling algorithm is as follows: For each citation in the collection, the algorithm tests if the citation really belongs to the given collection; First, remove an author from the citation; Then, guess back the removed author name using additional information. If the guessed name is not equivalent to the removed name, the citation is false citation.

In general, the of person name disambiguation has been studied in the domain of citations. Recently Bekkerman et al. [4] proposed techniques to disambiguating collections of Web appearances using Agglomerative and Conglomerative Double Clustering.

Chapter 3

The Split Entity Resolution (SER) Problem

In this chapter, we consider the Split Entity Resolution (SER) problem caused by duplicate entities (i.e., ambiguous author names) in bibliographic DLs (e.g., DBLP [47], CiteSeer [49], or e-Print arXiv [2]). These DLs have been an important resource for academic communities since scholars often try to search for relevant work from DLs. Researchers also use citation records in order to measure a publication's impact in the research community. It is thus an important task to keep the citation records consistent and up-to-date. However, because of various reasons (e.g., data-entry errors, diverse formats, and imperfect citation gathering softwares), citations of the same scholar commonly appear under different *name variants* in DLs. Thus, keeping citations correct and up-to-date proved to be a challenging task in a large-scale DL.

In general, the name variants refer to the different spellings of author names that are in fact referring to the same scholar. Because of this problem, it is difficult to get the complete list of the publications of some authors. For instance, imagine a scholar "Peter Keleher" has published 18 articles. However, a DL keeps three separate purported author names, "P. Keleher", "Peter J. Keleher", and "Peter Keleher", each of which contains 5, 9, and 4 citations, respectively. In such a case, users searching for all the articles of "Peter Keleher" will get only 5, 9, or 4 of them. Similarly, any bibliometrical study would underestimate the impact of the author "Peter Keleher", splitting his share into "Peter Keleher", "P. Keleher", "Peter J.

	gid	eid	author	title	venue
1	215862	192021	a. l. cox h. lu p. keleher r. rajamony s. d...	software versus hardware shared memory implement...	international conference on computer architecture
2	215862	506384	a. l. cox d. b. johnson e. n. elnozahy j. k. b...	session distributed shared memory	acm sigops european workshop
3	215862	827982	p. keleher	light weight currency management mechanisms in d...	ride
4	215862	829043	p. keleher s. k. tripathi f. w. miller	general data streaming	rtss
5	215862	851663	p. keleher k. thitikamol	multi threading and remote latency in software dsms	icdcs

	gid	eid	author	title	venue
1	224465	238760	dejan perkovic peter j. keleher	online data race detection via coherency guarant...	operating systems design and implementation
2	224465	296828	peter j. keleher	tapeworm	operating systems design and implementation
3	224465	301345	peter j. keleher	decentralized replicated object protocols	annual acm symposium on principles of distribut...
4	224465	305157	peter j. keleher	symmetry and performance in consistency protocols	international conference on supercomputing
5	224465	305170	jeffrey k. hollingsworth kyung dong ryu peter...	mechanisms and policies for supporting fine graine...	international conference on supercomputing
6	224465	305172	dejan perkovic peter j. keleher	responsiveness without interrupts	international conference on supercomputing
7	224465	332800	peter j. keleher	a high level abstraction of shared accesses	acm transactions on computer systems tocs
8	224465	355190	dejan perkovic peter j. keleher	a protocol centric approach to on the fly race dete...	ieee transactions on parallel and distributed syst...
9	224465	359448	peter j. keleher ugur cetintemel	consistency management in deno	mobile networks and applications

	gid	eid	author	title	venue
1	224467	165150	alan l. cox peter keleher sandhya dwarkadas ...	evaluation of release consistent software distri...	international conference on computer architecture
2	224467	277597	kritchalach thitikamol peter keleher	per node multithreading and remote latency	ieee transactions on computers
3	224467	831101	peter keleher	performance of mobile single object replicatio...	srds
4	224467	831139	peter keleher	efficient distributed precision control in symmetr...	srds

Figure 3.1: Screen-shot of three name variants of "Peter Keleher" from ACM in MS SQL Server (citations of "P. Keleher" (top), "Peter J. Keleher" (middle), and "Peter Keleher" (bottom)).

Keleher", and "Peter Keleher" incorrectly. Figure 3.1 shows this problem well.

In order to cope with the SER problem, we introduce a scalable two-step framework, in which step 1 is to substantially reduce the number of candidates via blocking, and step 2 is to measure the distance of two names via co-author information. Through this framework, we comparatively study alternative approaches to identify and correct such name variants. We consider several alternatives in each step of the framework (e.g., use sampling in the first step and TF/IDF Cosine similarity in the second step), and empirically evaluate the effects of various combinations of such algorithms for various data sets. We conduct an extensive experimental study to validate our claims – four alternatives in the first step and seven alternatives in the second step are examined on four different data domains, a total of $4 \times 7 \times 4 = 112$ combinations.

3.1 Two-step SER Framework: Problem

Problem Definition. We formally define the *Split Entity Resolution* problem as follows:

Given two *long* lists of entities (i.e., author names), X and Y, for each entity x ($\in X$), find a set of entities, y_1, y_2, ..., y_n ($\in Y$) such that both x and y_i ($0 \leq i \leq n$) are variants of the same entity.

The baseline approach to solve the problem is to treat each author name as a "string", and perform all pair-wise string distance using some distance function[1], $dist(x, y)$:

```
for each name x (∈ X) do
    for each name y (∈ Y) do
        if dist(x, y) < φ then
            x and y are name variants;
```

Algorithm 1: Baseline

Since the baseline approach is prohibitively expensive to run for large DLs (because of its quadratic time complexity, $O(|X||Y|)$), there is a need for more *scalable* algorithms that are not dependent on the syntactic similarities of author-name strings.

3.2 Two-step SER Framework: Solution

Figure 3.2 illustrates our two-step SER framework. We use the following ideas to design our algorithm: (1) Instead of comparing author-name spellings to find author-name strings that refer to the same author, we use information associated with the author-name strings like co-author list, authors' paper titles, venue list that authors often publish, or even institute etc. For instance, to identify if "Qiang Zhu" is the name variant of "Q. Zhu", instead of computing the string edit-distance of two names, we may test if there is any correlation between the co-author lists

[1] Both distance and similarity functions quantify closeness between two strings. That is, $0.0 \leq distance(l', r') \leq 1.0$ or $similarity(l', r') \leq 1.0$, where l' and r' are strings. In general, a smaller value indicates greater similarity between two strings in distance functions (e.g., Normalized Euclidean distance) while a larger value indicates greater similarity in similarity functions (e.g., Jaccard). In this paper, we will use these terms interchangably, depending on which interpretation is most natural. [14]

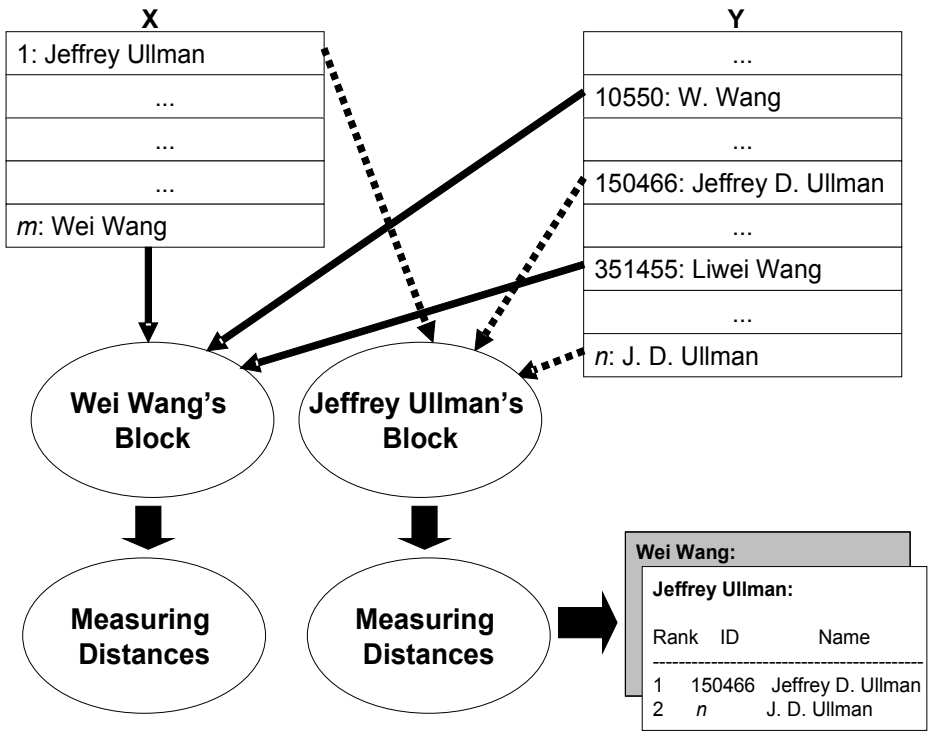


Figure 3.2: Overview of two-step framework.

of "Qiang Zhu" and "Q. Zhu." In the remainder of this chapter, we only focus on exploiting co-author information as the associated information of an author. Exploiting other associated information (or even hybrid of them as in [31]) is an interesting direction for future work (the case of exploiting both author name and its co-author is discussed in Section 3.4.1); (2) To make the algorithm scalable, we borrow the *Blocking* technique popular in the solutions of record linkage problem, and modify the baseline approach as follows:

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

3.2.1 Step 1: Blocking

The goal of step 1 is to put similar inputs into the same group by some criteria (thus called *Blocking*) so that distance measures of step 2 can be done per group. If the filtering criteria (i.e., blocking methods) are too aggressive, then only small

```
// let C_a be co-author information of author a;
for each name x (∈ X) do
    create a block B_x(∈ B);
// Step 1;
for each name y (∈ Y) do
    assign y to all relevant blocks B_i(∈ B);
// Step 2;
for each block B_x (∈ B) do
    for each name z (∈ B_x) do
        if dist(C_x, C_z) < φ then
            x and z are name variants;
```

Algorithm 2: Scalable Two-Step Algorithm

number of inputs will pass them and be put into the same block, and sometimes even right answers may be incorrectly pruned away. On the other hand, if the criteria are too lenient, then too many number of inputs (including noisy data) will pass them, bloating up the size of blocks. Furthermore, in general, the size of blocks has a close correlation with the scalability of the next step – the bigger a block is, the longer it takes to do step 2. In order to see the effects of different blocking schemes in the SER context, therefore, we examine four representative (and distinct) blocking methods – heuristic, token-based, n-gram, and sampling. Informally, given a set of p author names, n_1, ..., n_p, blocking methods return a set of q blocks, B_1, ..., B_q as follows:

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

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

3.2.1.1 Spelling-based Heuristics

The simplest approach is to group author names based on their name spellings, and can be attributed to a more general method known as *sorted neighborhood* [34]. In our context, all names with the same heuristics are grouped into the same block. For instance, "Jeffrey Ullman" is grouped together with "J. Ullman" if the heuris-

tics is "the same initial of the first name and the same last name (iFfL)". Other plausible heuristics are: "the same initial of the first name and the same initial of the last name (iFiL)," or "the same last name (fL)," or even the combination of the above. Different heuristics would have slightly different impact on the performance of the two-step methods. For instance, fL would generate bigger sized blocks than iFfL does so that it usually has a higher accuracy while being slower. Since comparing different heuristics is not the goal of this research, and iFfL is the most commonly used in citation matching context [37], in our experimentation, only iFfL is used.

3.2.1.2 Token-based

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

3.2.1.3 N-gram

The idea of the N-gram blocking is similar to that of the token-based one, except that it has finer granularity – instead of checking common tokens, this method checks the existence of common N continuous characters from author names (we use $N = 4$ that gave good results in [8]). Since the granularity is finer, the number of author names put into the same block is the largest among the four blocking methods. For instance, using the N-gram blocking, "**Davi**d R. Johnson" and "F. Barr-**Davi**d" are grouped into the same block because of the common 4-gram "davi".

3.2.1.4 Sampling

Another alternative blocking method is to use *Sampling*. Given a name string x, suppose we want to draw a number of samples that are most similar to x, and

group them into the same block. If the sampling process is somehow fast while it generates accurate samples, then it can serve as a good blocking method. One of the state-of-the-art sampling techniques that satisfy both criteria (i.e., being fast and accurate) is the *sampling-based join approximation* method recently proposed by [29]. We adopt it to our context as follows: Imagine each token from all author names has an associated weight using the TF/IDF metric in Information Retrieval (IR) (i.e., common tokens in author names have lower weights while rare ones have higher weights). Then, each author name t is associated with its token weight vector v_t. Suppose that for each name t_q in an author name set R_1, we want to draw a sample of size S from another author name set R_2 such that the frequency C_i of name $t_i \in R_2$ can be used to approximate $sim(v_{t_p}, v_{t_i}) = \sigma_i$. That is, σ_i can be approximated by $\frac{C_i}{S} T_V(t_q)$, where $T_V(t_q) = \sum_{i=1}^{|R_2|} \sigma_i$. Then, put t_i into a block only if $\frac{C_i}{S} T_V(t_q) \geq \theta$, where θ is a pre-determined threshold. This strategy assures that all pairs of names with similarity of at least θ can be put into the same block with a desired probability, as long as the proper sample size S is given.

3.2.2 Step 2: Measuring Distances

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

3.2.2.1 The Supervised Methods

Han et al. [31] recently showed the effectiveness of two supervised methods when there are enough number of training data, the *Naive Bayes model* [50] and the *Support Vector Machines* [16], in a slightly different name disambiguation context.

The Naive Bayes Model (NBM). In this method, we use Bayes' Theorem to measure the similarity between two author names. For instance, to compute the similarity between "Gene H. Golub" and "Golub, G.," we estimate each co-author's probability of "Gene H. Golub" in terms of the Bayes rule in training, and then calculate the posterior probability of "Golub, G." with the co-authors'

probabilities of "Gene H. Golub" in testing. As shown in Figure 3.2, given a block corresponding to an author name x in X and x's candidate names y_i in Y ($i \in [1, k]$, where k is the total number of author names selected from Y), we calculate the probability of each pair of x and y_i and find the pair with the maximal posterior probability as follows:

- **Training:** all co-author names of x are randomly split, and only the half is used for training. We estimate each co-author's conditional probability $P(A_m|x)$ conditioned on the event of x from the training data set, $A_i \in \{A_1, ..., A_j, ..., A_m\}$ and A_j is the j-th co-author of x:

$$P(A_j|x) = P(A_j|Frequent, CoAuthor, x) \times P(Frequent|CoAuthor, x) \times P(CoAuthor|x) + P(A_j|Infrequent, CoAuthor, x) \times P(Infrequent|CoAuthor, x) \times P(Alone|x)$$

  - $P(A_j|Frequent, CoAuthor, x)$: the probability of x working for a paper with a particular co-author A_j
  - $P(Frequent|CoAuthor, x)$: the probability of x working for a paper with the co-authors, who worked with x at least twice in the training data, conditioned on the event of x's previous co-authors
  - $P(CoAuthor|x)$: the probability of x working for a paper with co-authors in the future
  - $P(Alone|x)$: the probability of x writing a paper alone

- **Testing:** we use the following target function: $V_{NBM} = argmax_{y_i \in N}\{P(y_i)\Pi_k P(A_k|y_i)\}$

  - N: the total number of author names, selected from Y, in the block
  - k: the k-th co-author in y_i, being the common co-author as x

The Support Vector Machines (SVM). SVM is one of the popular supervised classification methods. In our context, it works as follows:

- **Training:** all co-authors of an author in a block is transformed into vector-space representation. Author names in a block are randomly split, and 50% is used for training, and the other 50% is used for testing. Given training

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

Table 3.1: Terms.

examples of author names labeled either *match* (e.g., "J. Ullman" and "Jeffrey D. Ullman") or *mismatch* (e.g., "J. Ullman", "James Ullmann"), SVM creates a maximum-margin hyperplane that splits the *match* and *mismatch* training examples.

- **Testing:** given SVM classifies vectors by mapping them via kernel trick to a high dimensional space where the two classes of equivalent pairs and different ones are separated by a hyperplane, and the corresponding similarity is obtained. For the SVM prediction, we use the Radial Basis Function (RBF) kernel [16], $K(x_i, y_i) = e^{-\gamma||x_i - y_i||^2}, (\gamma > 0)$, among alternatives (e.g., linear, polynomial, sigmoid kernels).

3.2.2.2 The Unsupervised Methods

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

String-based Distance. In this scheme, the distance between two author names are measured by the "distance" between their co-author lists. That is, to measure the distance between "Timos K. Sellis" and "Sellis, T. K.," instead of computing the sim("Timos K. Sellis", "Sellis, T. K."), we compute the sim(co-author-list("Timos K. Sellis"), co-author-list("Sellis, T. K.")). Among many possible distance measures, we use two token-based string distance metrics (e.g., *Jaccard* and *TF/IDF*), and two edit-distance based ones (e.g., *Jaro* and *Jaro-Winkler*) that were reported to give a good performance for the general name matching problem in [14]. We briefly describe the metrics below. For details of each metric, refer to [14].

Using the terms of Table 3.1, the four metrics can be defined as follows:

- **Jaccard**$(\mathbf{x}, \mathbf{y}) = \frac{|T_x \bigcap T_y|}{|T_x \bigcup T_y|}$
- **TF/IDF**$(\mathbf{x}, \mathbf{y}) = \sum_{w \in T_x \cap T_y} V(w, T_x) \times V(w, T_y)$
  - $V(w, T_x)$=$log(TF_{w,T_y} + 1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'}(log(TF_{w,T_y}+1) \times log(IDF_w))}}$ (symmetrical for $V(w, T_y)$)
  - TF_{w,T_x}: the frequency of w in T_x
  - IDF_w: the inverse of the fraction of names in a corpus containing w
- **Jaro**$(\mathbf{x}, \mathbf{y}) = \frac{1}{3} \times (\frac{|CC_{x,y}|}{|C_x|} + \frac{|CC_{y,x}|}{|C_y|} + \frac{|CC_{x,y}| - X_{CC_{x,y}, CC_{y,x}}}{2|CC_{x,y}|})$
- **Jaro** $-$ **Winkler**$(\mathbf{x}, \mathbf{y}) = Jaro(x, y) + \frac{max(|L|,4)}{10} \times (1 - Jaro(x, y))$
  - L: the longest common prefix of x and y

Vector-based Cosine Similarity. In this approach, instead of using string distances, we use vector space to measure the similarity of the co-author lists. We model the co-author lists as vectors in the vector space, each dimension of which corresponds to a unique author name appearing in the citations in the block. For example, suppose we have a block that has three groups of citations, one for "A. Y. Halevy," another for "Alon Halevy," and the other for "A. Halevy." In the first group, suppose we have five papers, each co-authored by "A. Y. Halevy" with one or more of "A. Doan," "L. Dong," and "M. Liu." Further suppose among the five papers "A. Y. Halevy" co-authored three times with "A. Doan," four times with "L. Dong," and once with "M. Liu." Similarly, suppose we have 10 papers in the second group, in which "Alon Halevy" co-authored five times with "A. Doan," seven times with "L. Dong," and three times with "P. Mork." In the last group, suppose we have seven papers in which "A. Halevy" co-authored three times with "P. Chopra," five times with "J. Xu," and once with "A. Doan."

The total number of unique co-author names in the block is 6 except the three "A. Y. Halevy" name variants. In order to determine if the three name variants are indeed referring to the same scholar, we model the co-author information for each variant as a vector and compute the similarities between the vectors. The resulting vectors have 12 dimensions, each of which corresponds to one of 6 unique co-author names appearing in the block. For example, for the first group of citations for "A.

Data set	Domain	# of authors # of citations	# of co-authors per author average median std-dev	# of tokens per author average mediane std-dev
DBLP	CompSci	364,377 562,978	4.9 2 7.8	11.5 6 18
e-Print	Physics	94,172 156,627	12.9 4 33.9	33.4 12 98.3
BioMed	Medical	24,098 6,169	6.1 4 4.8	13.7 12 11
EconPapers	Economics	18,399 20,486	1.5 1 1.6	3.7 3 4.1

Table 3.2: Summary of data sets.

Y. Halevy," we have a vector v("A. Y. Halevy") = [0 3 4 1 0 0], provided that the dimensions are ordered by co-authors' last name and values in the vector represent the number of papers that the author represented by the dimension co-authored with "A. Y. Halevy." For example, the value, 1, in the third dimension represents the number of papers that "M. Liu" co-authored with "A. Y. Halevy." Similarly, we have v("Alon Halevy") = [0 5 7 0 3 0] and v("A. Halevy") = [3 1 0 0 0 5]. In order to measure the similarity between the vectors, $\vec{v}$ and $\vec{w}$, we use the simple Cosine similarity, an angle between two vectors, defined as: $\cos\theta = \frac{\vec{v}\cdot\vec{w}}{\|\vec{v}\|\cdot\|\vec{w}\|}$.

3.3 Experimental Set-up

Data sets. We have gathered real citation data from four different domains, as summarized in Table 3.2. Compared to previous work, all of the four data sets are substantially "large-scale" (e.g., DBLP has 360K authors and 560K citations in it). Different disciplines appear to have slightly different citation policies and conventions as shown in Table 3.3. For instance, Physics and Medical communities seem to have more number of co-authors per article than Economics community.

Stephan Wiesener, Wolfgang L. J. Kowarschick, Pavel Vogel, Rudolf Bayer: Semantic Hypermedia Retrieval in Digital Libraries. ADL 1995: 115-12

(a) DBLP

Optical and Infrared Properties of the 2 Ms Chandra Deep Field North X-Ray Sources. A. J. Barger, L. L. Cowie, P. Capak, D. M. Alexander, F. E. Bauer, E. Fernandez, W. N. Brandt, G. P. Garmire, and A. E. Hornschemeier. The Astronomical Journal, 126, 632-665 (2003)

(b) e-Print

Luiz F Poli de Figueiredo, Mali Mathru, Jaclyn R Jones, Daneshvari Solanki, George C KramerInhaled nitric oxide reverses cell-free hemoglobin-induced pulmonary hypertension and decreased lung compiance, crit care 1997 1: 111-116.

(c) BioMed

"The Choice of IPO Versus Takeover: Empirical Evidence," James C. Brau, Bill Francis and Ninon Kohers, Journal of Business Volume 76, Issue 4, Pages 583-612, 2003

(d) EconPapers

Table 3.3: Example citations of each data set.

Furthermore, the conventions of citations also vary. For instance, citations in e-Print use the first name of authors as only initial, while ones in DBLP use full names.

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

To make solution sets, for each data set, we prepare two lists of author names, X and Y, where X is initially empty, and Y contains the entire author names (e.g., 364,377 names for DBLP). Then, we pick top-100 author names from Y according to their number of citations, and generate 100 corresponding new name variants artificially. Note that the reason why we have to use "top 100" instead of "random 100" is because of two supervised methods in step 2. That is, two supervised methods, first proposed in [31], do not work without enough numbers of training sets. For instance, for "Grzegorz Rozenberg" with 344 citations and

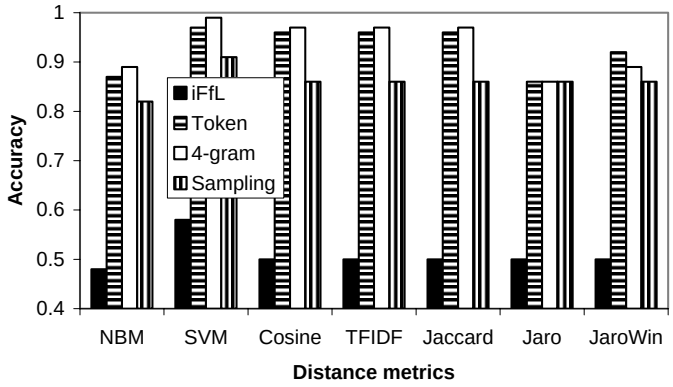


Figure 3.3: Accuracy with various error types (DBLP).

114 co-authors in DBLP, we create a new name like "G. Rozenberg" (abbreviation of the first name) or "Grzegorz Rozenbergg" (typo in the last name). Then, after splitting the original 344 citations into halves, each name carries half of citations, 172, and is put back into X and Y, respectively. At the end, there are 100 and 364,377 names in X and Y, respectively. Then, through the proposed two-step framework, for each name in X, we test if the algorithm can find the corresponding artificial name variant in Y (that we generated and thus know what they are).

Note that the way we generate artificial name variants may affect the performance of blocking. For instance, if all artificial names that we generated are abbreviation of first names, then both heuristic and 4-gram blocking would work well. However, if artificial name variants were generated by adding garbage characters in last names, then this will negatively affect the performance of heuristic blocking, while it has little effect on the 4-gram blocking method. In general, it is difficult to precisely capture the right percentages of different error types in author name variants. For the original name "Ji-Woo K. Li", some of possible error types are name abbreviation ("J. K. Li"), name alternation ("Li, Ji-Woo K."), typo ("Ji-Woo K. Lee" or "Jee-Woo K. Lee"), contraction ("Jiwoo K. Li"), omission ("Ji-Woo Li"), or combinations of these.

To quantify the effect of error types on the accuracy of algorithms in the framework, we first compared two cases: (1) mixed error types of abbreviation (30%), alternation (30%), typo (12% each in first/last name), contraction (2%), omission (4%), and combination (10%); and (2) abbreviation of the first name (85%) and

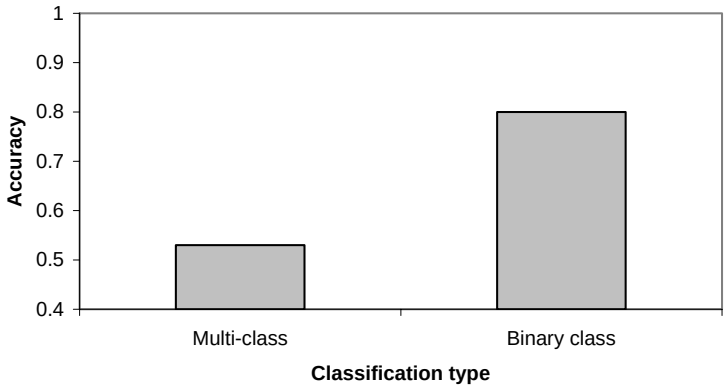


Figure 3.4: Comparison of multi-class vs. binary-class classification based SVM.

typo (15%). The accuracy of the former case is shown in Figure 3.3, and that of the latter case is in Figure 3.10(a). Note that regardless of the error types or their percentages, all blocking methods, except iFfL, show reasonably similar accuracies. That is, since both token-based and sampling-based blocking have a granularity of "tokens" in processing, they are more tolerable than iFfL which can only handle well name abbreviation error type. Likewise, since the granularity of 4-gram is the finest (i.e., characters), it shows the best tolerance for diverse error types. All subsequent experimentations are done using the latter case (85%/15%).

Implementations. We have implemented various algorithms of Section 3.2.1 and 3.2.2 as follows: (1) *Step 1*: Token-based and N-gram blocking methods are implemented by the open source tool, SecondString [65], where all space-delimited words from co-author lists are treated as tokens and "4" was used for N-gram (i.e., 4 continuous characters) as tokens. For the sampling-based blocking, we used the implementation of [29] with a sample size of 64 and a threshold of 0.1, and ran the experimentation in Microsoft SQL Server 2000. (2) *Step 2*: For the supervised learning methods, citations per author are randomly split, with half of them used for training, and the other half for testing. For the implementation of Support Vector Machines, LIBSVM [25] was used. In particular, we found that the multi-class classification based implementation in [31] performs poorly for our experimentation. This is because a classifier needs to be prepared for each candidate, and there are usually large number of candidates in our large-scale setting.

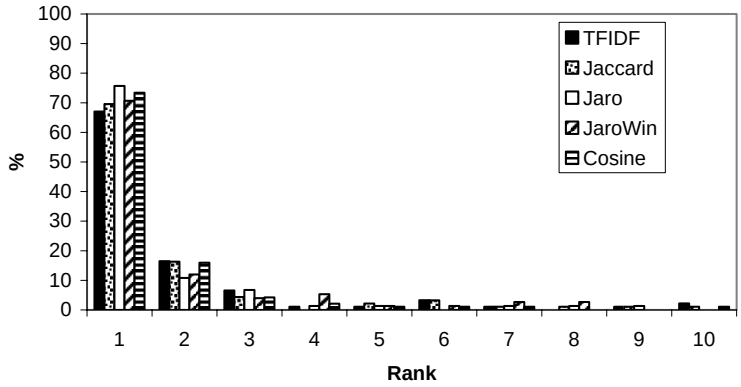


Figure 3.5: Distribution of name variants in top-10.

Therefore, instead, our implementation of SVM is based on binary-class classification. In the comparison, as shown in Figure 3.4, binary-class classification based SVM showed an accuracy about 25% higher than multi-class classification based one (while taking about 20% less time). For the string-based distance functions of the unsupervised learning methods, we used the implementations of TF/IDF, Jaccard, Jaro, and Jaro-Winkler from SecondString [65].

Evaluation metrics. Two main metrics that we used are *scalability* and *accuracy*: (1) The scalability of the framework was measured by the "size" of blocks generated in step 1 and the "time" it took to process both step 1 and 2; (2) To measure how effectively name variants can be found, we measured the "accuracy" of top-k as follows. For a name in X, our algorithm finds top-k candidate name variants in Y. If the top-k candidates indeed contain the solution, then it's *match*. Otherwise, it is a *mis-match*. This is repeated for all 100 names in X. Then, overall accuracy is defined as: Accuracy $= \frac{\#\text{ of matches}}{100}$.

The accuracy was measured for different k values (i.e., $k = 1, 5, 10$). For instance, with $k = 5$ in the DBLP data set, for each author in X, methods return the top-5 candidate name variants out of 364,377 authors, and if one of these 5 candidates is the artificial name variant that we created, then it is a match. We repeated all of the subsequent experiments for three window sizes of 1, 5, and 10, and found that accuracies with larger window size ($k = 10$) are about 10% higher than those with smaller window size ($k = 1$). The seemingly only-small drop of

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

Table 3.4: Solution space.

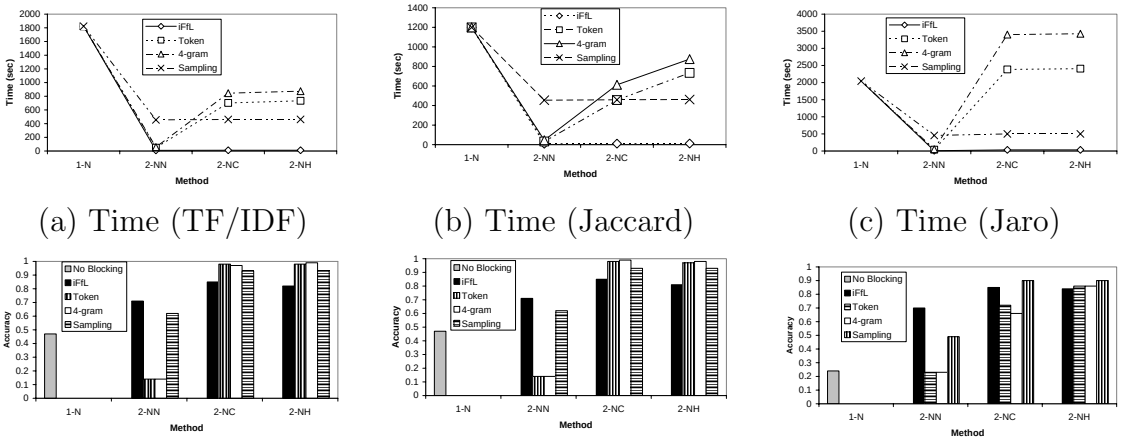
the accuracy can be explained as follows. As shown in Figure 3.5, about 70% of correct name variants are returned as the first among 10 candidates. That is, even if we use a smaller window size ($k = 1$), 70% of name variants would be found correctly. Since the most of name variants are found within rank 3, when we used $k = 5$, its accuracy is almost as good as that of $k = 10$. In the following, therefore, we show the results for $k = 5$.

3.4 Experimental Results

3.4.1 Our Framework versus Other Alternatives

Table 3.4 summarizes four possible approaches to the given problem: (1) 1-N is a single-step pair-wise name matching scheme without using blocking or co-author information; (2) 2-NN uses the two-step approach, but do not exploit co-author information; (3) 2-NC is the main proposal in the framework; and (4) 2-NH is the modification of 2-NC in that in step 2, it combines both author and co-author information together with proper weights (e.g., we used 1/4 and 3/4 for author and co-author, respectively).

Figure 3.6 summarizes the experimental results of four alternatives using three representative metrics – TF/IDF, Jaccard, and Jaro. In terms of the processing time, 1-N is the slowest for TF/IDF and Jaccard, as expected, due to its quadratic time complexity (i.e., $100 \times 364,377$ times of pair-wise name comparisons). However, Figure 3.6(c) shows a rather unexpected result in that both 2-NC and 2-NH take more time than 1-N does, despite their advantages through blocking in step 1. This is due to the combination of the slow distance computation of Jaro method and long co-author list, and can be explained as follows. To make the discussion simple, suppose computing $dist(x, y)$ takes 1 sec, and its computation time is pro-



(a) Time (TF/IDF) (b) Time (Jaccard) (c) Time (Jaro)

(d) Accuracy (TF/IDF) (e) Accuracy (Jaccard) (f) Accuracy (Jaro)

Figure 3.6: Comparison of four alternatives (DBLP with $k = 1$).

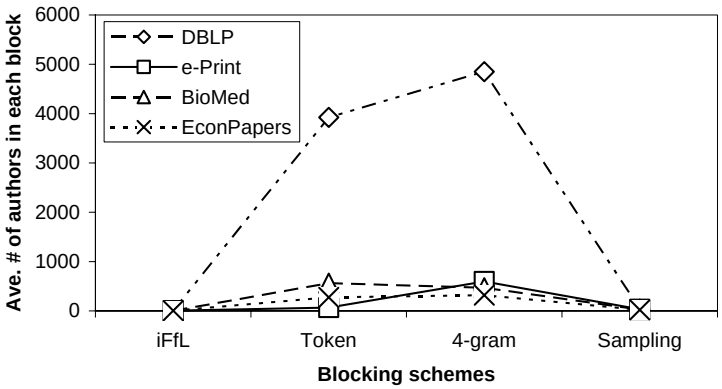


Figure 3.7: Average # of authors per block.

portional to the number of characters in x and y. That is, $dist(x, y)$ takes 1 sec, while $dist(x^{100}, y^{20})$ takes $100 \times 20 = 2,000$ sec (if x^n denotes characters x with the length of n).

Then, (1) Since 1-N computes $dist(x, y)$ for all pairs from X and Y, it takes $|X| \times |Y| = 100 \times 364,377 \approx 36.4M$ sec. (2) In Section 3.3, we chose top-100 authors with the most number of citations as our targets. Since these 100 authors have a large number of citations, their number of co-authors is large too (i.e., on average 100 co-authors per author). Therefore, using 2-NC, computing $dist(x', y')$ (i.e., x' and y' are co-authors of x and y, respectively) is equal to computing

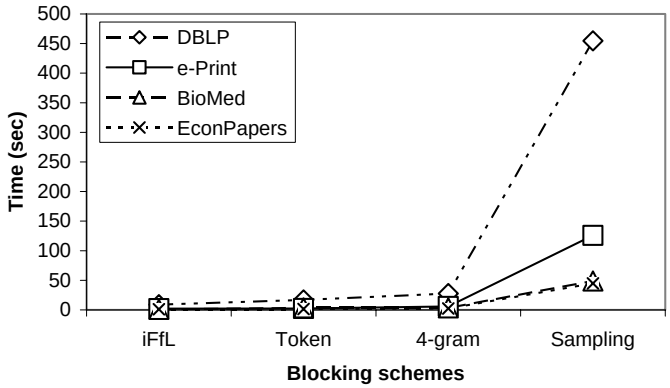
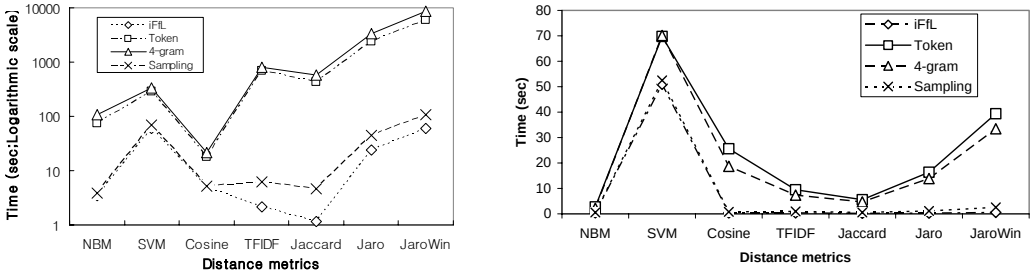


Figure 3.8: Processing time for Step 1.

$dist(x^{100}, y^{100})$, taking about $100 \times 100 = 10,000$ sec. Since all names in a block must be compared, if a block has 3,500 names (e.g., 4-gram blocking in Figure 3.7), then it takes $3,500 \times dist(x', y') = 3,500 \times 10,000 \approx 35M$ sec. Note that the time for 1-N, $36.4M$ sec, is roughly the same as that for 2-NC, $35M$ sec. That is, when computation-heavy distance metrics such as Jaro or Jaro-Winkler are used in step 2, and since co-author names to consider are very long, the expense offset the benefit of the blocking in step 1. Note that our 100 target names in testing are those with the largest number of co-authors. Therefore, the scenario in our experimentation is the "worst case" for 2-NC.

In terms of accuracy, both 2-NC and 2-NH shows about 20%-30% improvement, compared to 1-N and 2-NN, validating the assumption that exploiting additional information is more beneficial than the simple name spelling in the SER problem. Compared to 2-NC, 2-NH shows no significant improvement. However, for Jaro method (Figure 3.6(f)), the accuracy of 2-NH, when token-based or 4-gram blocking is used, improves by 10%-15% from 2-NC. Note that Jaro method tends to work better when an input string is short. Therefore, when both name and co-author are considered in 2-NH, Jaro takes advantage of relatively good accuracy from "name" side, although it suffers from "co-author" side. At the end, this results in about 10%-15% improvements of accuracy. Since 2-NH takes longer than 2-NC while it shows only a negligible improvement in accuracy, in the remaining experiments, we use 2-NC as our default scheme.



(a) DBLP (b) EconPapers

Figure 3.9: Processing time for Step 2.

3.4.2 Scalability

Figure 3.7 illustrates the average number of authors per each block. Regardless of data sets, 4-gram blocking generates the most number of author names into each block. On the other hand, both heuristic and sampling blocking methods put a small number of author names into each block. Nevertheless, the time it takes to perform the blocking is quite small, except the sampling-based blocking which needs a pre-processing step for TF/IDF weighting and sampling as shown in Figure 3.8.

The processing time for step 2 is shown in Figure 3.9 for DBLP (364,377 authors) and EconPapers (18,399 authors) data sets. The other two data sets have similar graphs to that of EconPapers and omitted. In general, Cosine similarity is the fastest and edit-distance based metrics such as Jaro or Jaro-Winkler are the slowest. This is especially so since the string to compare is a long co-author list (instead of short author name). Both NBM and SVM are relatively faster than TF/IDF, Jaccard, Jaro, and Jaro-Winkler. Note that those token-based distance metrics such as TF/IDF and Jaccard are slower than NBM, SVM, and Cosine similarity methods, because there are a large number of candidate name variants in each block. Detailed comparison of cosine similarity vs. two supervised methods (NBM and SVM) are shown in Table 3.5. Although all three appear to take the same processing time in Figure 3.9 due to the enlarged Y-axis, cosine method is in fact much faster than the others, showing a better scalability. The SVM takes more time than the others since the hyperplane needs to be split in succession due to SVM's binary-class classifier.

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

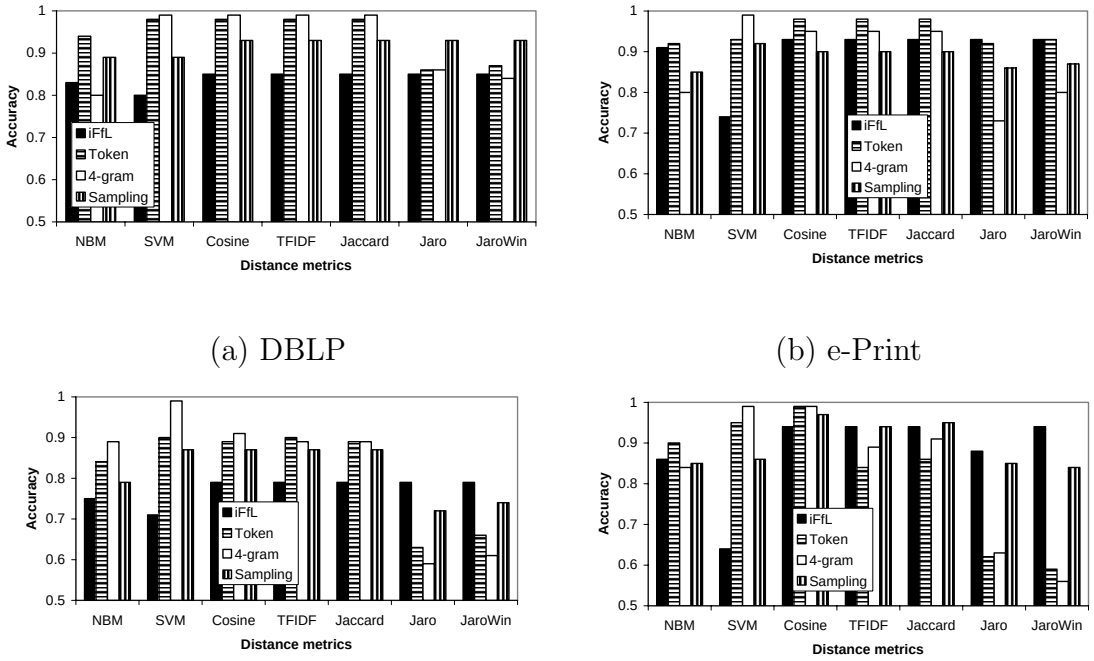
Table 3.5: Processing time for step 2 of NBM, SVM, and Cosine methods.

3.4.3 Accuracy

Figure 3.10 summarizes the accuracies of four blocking methods of step 1 combined with seven distance metrics of step 2 for all four data sets (with $k = 5$). Several phenomenons are noticeable.

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

The accuracies of DBLP and e-Print data sets are better than that of BioMed (and the omitted EconPapers) data set. The poor performance of BioMed case is mainly due to the small number of citations per author in data set. Since 2-NC scheme is exploiting co-author information of the author in question to find name variants, the existence of "common" co-author names is a must. However, in the BioMed data set, each author has only a small number of citations, 1.18, on average, and only small number of co-authors, 6.1, on average, making a total number of co-authors as $7.19 = 1.18 \times 6.1$ (assuming all co-authors are distinct). Therefore, for two arbitrary author names x and y, the probability of having "common" co-authors in the BioMed data set is not high. On the other hand, for the e-Print data set, the average number of citations (resp. co-authors) per



(a) DBLP

(b) e-Print

(c) BioMed

(d) EconPapers

Figure 3.10: Accuracy comparison ($k = 5$).

author is higher, 4.27 (resp. 12.94), making the total number of co-authors as $55.25 = 4.27 \times 12.94$ – roughly 8 times of the BioMed data set.

In general, Jaro or Jaro-Winkler method in step 2 gave poorer accuracy than the others. Since they are edit-distance based methods that are heavily affected by the number of transpositions, as the length of string to compare increases (in 2-NC, it is a long co-author string), its error rate increases as well. In the e-Print data set, the accuracies are lower, compared to those of DBLP, when the sampling-based blocking is used in step 1. This is because most citations in the e-Print data set use abbreviation for the first name of authors (e.g., "F. E. Bauer" or "E. Fernandez" in Table 5.2). Since the sampling technique use TF/IDF for weighting tokens, common tokens like abbreviated first name (e.g., "E.") would have lower weight via IDF, negatively affecting matching process. This is why the sampling-based blocking performed worse than the plain token-based blocking.

3.4.4 Summary of Experiments

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

3.5 Summary

Based on our scalable two-step framework, we compared various configurations – four blocking in step 1 (i.e., heuristic, token, N-gram, sampling), seven distance metrics via "co-author" information in step 2 (i.e., NBM, SVM, Cosine similarity, TF/IDF, Jaccard, Jaro, Jaro-Winkler), against four data sets (i.e., Computer Science, Physics, Medical, Economics). Experimental results verify that our proposed two-step framework using co-author relation (instead of author name alone) shows much improved scalability and accuracy (e.g., 4 times faster and 50% more accurate using sampling and TF/IDF on DBLP data set) compared to one-step approach.

Chapter 4

The Grouped Entity Resolution (GER) Problem

In this chapter, we focus on the Grouped Entity Resolution (GER) problem, where each *grouped entity* has "a group of elements" in it. Examples include authors with a paper list or actors with a movie list. Unlike the previous approaches, relying on textual similarity and producing a large number of false positives [1], we presented the experience of applying a recently proposed graph mining technique, *distQC*, atop conventional entity resolution solutions. This approach exploits contextual information mined from the group of elements per entity in addition to syntactic similarity. Furthermore, we focus on the intuition that two groups can be linked to each other if there is a high enough simialrity between matching pairs of individual records that constitute the two groups and there is a large fraction of such matching record pairs. To formalize this intuition, we present a *Group Linkage* measure based on bipartite graph matching with better accuracy than the existing textual similarity measures.

4.1 Quasi-Clique based Distance Measure

With an array of extensive research on the ER problem (surveyed in Chapter 2), in general, there are various efficient and effective methods to identify split entities.

[1] An entity determined to be a variant when it is not.

However, we observed that a particular type of entities occur quite common in real applications, and a more aggressive method can exploit the characteristics of the type better. That is, we noticed that many entities contain "a group of elements" in them. We refer to such an entity as a **Grouped-Entity** and the ER problem on grouped-entity as the **Grouped-Entity Resolution (GER)** problem. Unlike a regular entity, a grouped-entity contains a wealth of information in its elements. How to unearth such hidden information for resolving grouped-entities is the focus of this problem. Throughout the rest of this chapter, we simply use "entity" to refer to the grouped-entity.

The GER problem frequently occurs in many situations as follows:

- Examples of grouped-entities include an author entity with a paper list or an actor entity with a movie list. Figure 1.1 and Figure 1.6 illustrate the screen-shots of the GER problem from two real digital libraries – ACM and DBLP. Here, each grouped-entity (i.e., author) has a group of citations in it. Due to various errors, however, citations of the same author may be split under multiple author names. For instance, in the ACM, the citations of computer scientist "Jeffrey D. Ullman" appear under ten different name variants. If we take entity "Jeffrey D. Ullman" as the canonical one, then the other nine (e.g., "D. Ullman" and "J. Ullmann") are variants, and should be consolidated. Similarly, in DBLP, a partial list of citations of "Yin-Feng Xu" appears under the variant entity "Yinfeng Xu."

- The US census bureau[2] needs to keep track of people in families. To do this, they often use people's names or known addresses. However, due to data-entry errors or confusing homonyms, the tracking is not always successful. Moreover, comparing two people by their names alone yields many false positives. Now, suppose one uses as *context* the family information of each person – two persons are similar if their "families" are similar, e.g., they share the similar family members such as names of spouse and children. That is, each grouped-entity (i.e., person) has a group of elements (i.e., family names). Then, in determining if "John Doe" is the same person of "Jonathan Doe," for instance, one may combine the textual similarity, sim("John Doe",

[2]The example is derived from the discussion with Divesh Srivastava at AT&T Labs.

"Jonathan Doe"), as well as the contextual similarity – if the family of "John Doe" shares similar names with that of "Jonathan Doe." If so, it is likely that "Jonathan Doe" is a name variant of "John Doe."

By and large, previous approaches to the GER problem (e.g., [8, 34, 11]) work as follows: (1) the information of an entity, e, is captured in a data structure, $D(e)$, such as a multi-attribute tuple or an entropy vector; (2) a binary distance function, f, is prepared; (3) the distance of two entities, e_1 and e_2, is measured as that of the corresponding data structures, $D(e_1)$ and $D(e_2)$, using function f: $dist(e_1, e_2) = f(D(e_1), D(e_2))$; and (4) finally, if the result, $r = dist(e_1, e_2)$, is less than certain threshold, ϕ, then the two entities are variants: $r < \phi \rightarrow e_1 \sim e_2$. Although it works well in many scenarios, this approach often suffers from a large number of *false positives* (i.e., an entity determined to be a variant when it is not). Consequently, the overall recall and precision suffer. If a user asks for top-k answers, such false positives can even override correct variants out of the answer window of $|k|$, degrading the precision substantially.

Consider the example of Figure 1.1 again. Suppose the distance of two entities is measured as that of author name spellings themselves. If we use Edit distance, for instance, $dist$("J. D. Ullman", "J. Ullman") = 2. If the threshold ϕ is set to 3, then "J. Ullman" can be correctly identified as a variant of "J. D. Ullman". However, other entities such as "J. L. Ullman" or "K. Ullmann" whose distances are less than 3 will also be detected as variants. However, both are in fact *false positives*. Even if we use more sophisticated data structures to capture entities, the problem may still persist. For instance, suppose an author entity of Figure 1.1 is represented as a multiple-attribute tuple, `(coauthor, title, venue)`, where each attribute is a vector of tokens from the citations. That is, "J. D. Ullman" and "K. Ullman" are represented as `([Hopcroft, Aho], [Cube, Query], [KDD, ICDM])` and `([Cruise, Kidman], [Mining, Query], [KDD, SIGMOD, ICDM])`, respectively. Then, depending on the distance function, it is possible that "K. Ullman" is identified as a variant of "J. D. Ullman" since both share many tokens. However, "K. Ullmann" is a false positive which happens to bear certain textual similarity to "J. D. Ullman" (since maybe both share research interests).

The culprit of this false positive problem is in essence the use of distance func-

tions that solely rely on the "textual similarity" of two entities, regardless of the adopted data structures. Toward this problem, in this chapter, we present a novel graph partition based approach that boosts up precision significantly. We unearth the relationship hidden under the grouped-entities, and exploit it together with textual similarities. Experimental results using real and synthetic data sets validate our claim.

Our contributions are as follows:

- We formulate the GER problem as a special form of the ER problem. Since the grouped-entities in the GER problem contain a wealth of information (i.e., a group of elements), the exploitation of GER can result in better outcome.
- We introduce how to capture "contextual information" hidden in a group of elements in grouped-entities. In particular, we propose to use the technique of *superimposition* to mine hidden relationships into graphs.
- To capture the "contextual distance" between grouped-entities, we exploit the notion of *Quasi-Clique*, a measure to see how strong inter-relationships between two graphs are, and propose a simple yet effective two-step GER algorithm, **distQC**.
- The proposed techniques of superimposed graphs and subsequent *Quasi-Clique*-based distance measurement are implemented and tested extensively against five test cases (1 real and 4 synthetic cases). Experiments verify that our proposal improves precision and recall up to 83% (when used with a variety of existing ER solutions) but never worsens them.

4.1.1 Quasi-Clique based Distance Measure: Problem

Unlike regular entities, each grouped entity contains *a group of tuples* in it. Examples include an author entity with a paper list or an actor entity with a movie list. This is illustrated in Figure 4.1(a). Then, the distance of two entities can be measured by many things. For instance, we compare "Jeffrey D. Ullman" and "Jeffrey Ullman" to determine if both author entities are variants or not (Figure 4.1(b)).

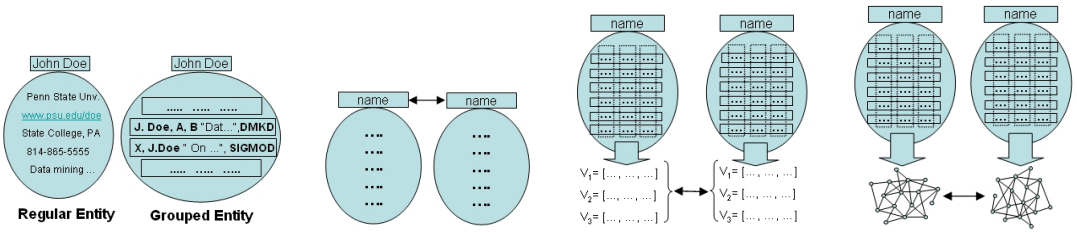


Figure 4.1: Illustration of our approach: (a) regular vs. grouped entities; (b) distance based on entity names; (c) distance based on the contents of entities (as multi-attribute vectors); and (d) distance based on the context of entities (as graphs).

Similarly, two address entities, e_1 and e_2, may be detected as variants if their corresponding properties, ("Foster Ave.", 16802, "PA") and ("East Foster Ave.", 16802, "Pennsylvania"), are similar. For the case of grouped entities, one can represent a group of tuples in some data structures, and measure the distance as the distance of two data structures. For instance, Figure 4.1(c) illustrates when vectors used to capture the contents of grouped entities (i.e., multi-attribute tuples). Typically, the group of tuples in the grouped entities follows certain formats, and contains a lot of tokens "not" all of which are directly indicative of the entity. For instance, the grouped entity of "Yin-Feng Xu" of Figure 1.6 contains 19 citations, each of which contains co-authors, title, venue, year, etc. Then, some tokens in this long list are not real indicative of the entity "Yin-Feng Xu" and may confuse regular distance functions. Therefore, often, detecting variant grouped entities yields a large number of false positives. Now, we formally define the *Grouped Entity Resolution* problem as follows:

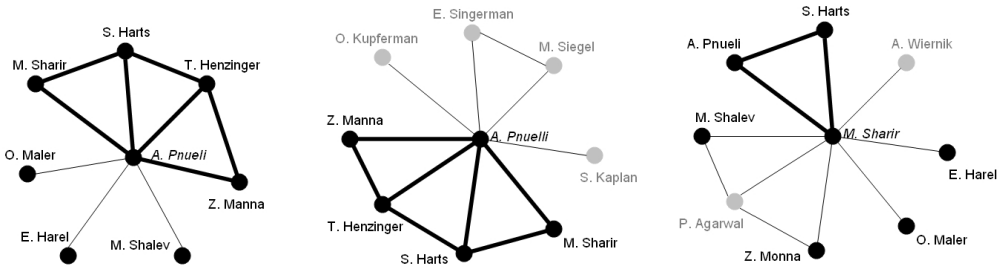
Given a set of grouped entities, E, where each entity contains a group of tuples, for each canonical entity, e_c ($\in E$), identify all variant entities, e_v ($\in E$), such that $dist(e_c, e_v) < \phi$ with as few false positives as possible.

Note that our proposal works as long as "one" entity is selected as the canonical one.

4.1.2 Quasi-Clique based Distance Measure: Solution

In general, as a domain independent technique, string distance metrics that compare values of entity attributes are mainly employed to determine if two entities are variants. However, we consider a web of entities connected via relationships present in data sets. Our approach is based on the hypothesis that "if two entities are variants, there is a high likelihood that they are strongly connected to each other through a web of relationships, implicit in the database" [40]. In other words, among "J. D. Ullman," "J. Ullman," and "J. K. Ullman" entities, in order not to mistakenly mark "J. K. Ullman" as a variant of "J. D. Ullman" (i.e., no false positive), we may unearth the hidden relationships between two and exploit them (e.g., although both are Database researchers, the cores of their frequent co-author list are slightly different). This is illustrated in Figure 4.1(d), where the contents of the grouped entities are first captured as some "graphs" and subsequently the distance between two graphs are used to measure the distance between two grouped entities. Since graphs may contain richer information than simple strings or vectors, the corresponding distance results from graphs are likely to help identify real variants.

Using Context as Additional Information. In general ER problem (e.g., record linkage, reference resolution, name disambiguation), often, the underlying assumption is that there are some textual similarities among variant entities. For instance, to identify if two given records are duplicate or not (i.e., record linkage problem), one may apply either a token-based distance function (e.g., Jaccard) or an edit-distance based function (e.g., Jaro) to measure how similar two records are. However, in the situations where matching entities may "not" bear syntactical similarities, the general ER methods do not work well. On the other hand, an interesting observation from the previous studies is that a method often has good performance if it can consider some "additional information" beyond textual similarities. We call this additional information as **context**. To illustrate the usefulness of context, let us consider the following example. First, the US census bureau needs to keep track of people in families. To do this, they often use people's names or known addresses. However, due to data entry errors or confusing homonyms, the tracking is not always successful. Moreover, comparing two persons by their names alone yields many false positives. Now, suppose we



(a) "A. Puneli" entity (b) "A. Punelli" entity (c) "M. Sharir" entity

Figure 4.2: Graph representations of the "contexts" (i.e., co-authors) of three grouped entities, and their superimposed quasi-cliques (solid lines).

use as *context* the family information of each person – two persons are similar if their "families" are similar (e.g., they share the similar family members such as names of spouse and children). Then, in determining if "John Doe" is the same person as "Jonathan Doe", for instance, one may combine the textual similarity, sim("John Doe", "Jonathan Doe"), as well as the contextual similarity – if the family of "John Doe" shares similar names with that of "Jonathan Doe." If so, it is likely that "Jonathan Doe" is a name variant of "John Doe," both of whom share the similar family members. As a second example, let us consider the collaboration network of digital libraries (i.e., a network with vertices being authors and edges being co-authorship relations). Look at the GER problem of Figure 1.1 again. Here, the goal is to determine if two author entities, say, "J. D. Ullman" and "J. K. Ullman", refer to the same person or not. Since each entity is a type of the "grouped" entity, it has a group of tuples in it. That is, each has abundant repetition of related tokens – a set of co-author names, tokens of words used in paper titles, or a set of venue names to which they submit often, etc. We may use these as *context* and measure the corresponding contextual similarity. Consider the following example drawn from the ACM digital library.

There are three grouped entities, E_a ("A. Puneli"), E_b ("A. Punelli"), and E_c ("M. Sharir"), of which E_b is the name variant of E_a and E_c is the false positive of E_a. In the ACM data set, each entity has the following co-authors in their group of tuples:

- E_a={"M. Sharir", "S. Harts", "T. Henzinger", "Z. Manna", "M. Shalev", "E. Harel", "O. Maler}

- E_b={"M. Sharir", "S. Harts", "T. Henzinger", "Z. Manna", "O. Kupferman", "S. Kaplan", "E. Singerman", "M. Siegel"}
- E_c={"A. Pnueli", "S. Harts", "Z. Monna", "A. Wiernik", "P. Agarwal", "M. Shalev", "E. Harel", "O. Maler"}

If we draw graphs where the entity itself becomes the center node and its co-authors become neighboring vertices attached to the center node, then we get Figure 4.2. Furthermore, if there are known co-authorships among co-authors, edges are created between them. For instance, "S. Harts" and "T. Henzinger" co-authored elsewhere (i.e., other than in entities E_a, E_b, and E_c), and thus an edge connecting them is created in Figure 4.2(a). First, suppose we compare three entities by counting how many common co-authors they have. Then, E_b and E_c have 5 and 7 common co-authors with E_a, respectively. Therefore, if we use distance metrics that are based on the number of common tokens of two entities such as Jaccard distance, then E_c would be probably returned as a variant of E_a over E_b. However, this is incorrect, and we have a case of false positive. Second, if we compare three entities in terms of how large the maximum common subgraph (where all vertices and adjacent edges match) is, then E_b and E_c have a common subgraph with 5 and 3 vertices with E_a, respectively. Therefore, E_b would be returned as a variant of E_a over E_c – the opposite result of previous case. Note that entities E_a and E_b have four common co-authors, {"M. Sharir", "S. Harts", "T. A. Henzinger", "Z. Manna"}, who are all well connected among themselves. This can be used as a clue to unearth the hidden similarity between two entities, E_a and E_b. On the other hand, E_c shares only small-sized well-connected subgraph although overall it has more number of common co-authors. In other words, if we look at the relationships existing in the whole graph, instead of individual vertices, then we may be able to detect that E_c is not a variant of E_a.

Based on this observation, in the following sections, we introduce our proposal of using graph-based partition to tackle the GER problem. We capture contextual information as graphs through superimposition, and measure their contextual similarity in terms of Quasi-Clique [56].

Quasi-Clique. Given a graph G, let $V(G)$ and $E(G)$ be the sets of vertices and edges in the graph, respectively. Let $U \subseteq V(G)$ be a subset of vertices. The

subgraph induced on U, denoted by $G(U)$, is the subgraph of G whose vertex-set is U and whose edge-set consists of all edges in G that have both endpoints in U, i.e., $G(U) = (U, E_U)$, where $E_U = \{(u, v) | (u, v) \in E(G) \wedge u, v \in U\}$. A connected graph G is a *γ-quasi-complete graph* ($0 < \gamma \leq 1$) if every vertex in the graph has a degree at least $\gamma \cdot (|V(G)| - 1)$. Clearly, a 1-quasi-complete graph is a complete graph (i.e., clique). In a graph G, a subset of vertices $S \subseteq V(G)$ is a *γ-Quasi-Clique* ($0 < \gamma \leq 1$) if $G(S)$ is a γ-quasi-complete graph, and no proper superset of S has this property. Clearly, a 1-Quasi-Clique is a clique. As shown in [56], an interesting property of Quasi-Cliques is that when γ is not too small, a γ-Quasi-Clique is compact – the diameter of the Quasi-Clique is small. Technically, let G be a γ-quasi-complete graph such that $n = |V(G)| > 1$. [56] proves the following result.

$$diam(G) \begin{cases} = 1 & \text{if } 1 \geq \gamma > \frac{n-2}{n-1} \\ \leq 2 & \text{if } \frac{n-2}{n-1} \geq \gamma \geq \frac{1}{2} \\ \leq 3\lfloor \frac{n}{\gamma(n-1)+1} \rfloor - 3 & \text{if } \frac{1}{2} > \gamma \geq \frac{2}{n-1} \text{ and} \\ & n \bmod (\gamma(n-1)+1) = 0 \\ \leq 3\lfloor \frac{n}{\gamma(n-1)+1} \rfloor - 2 & \text{if } \frac{1}{2} > \gamma \geq \frac{2}{n-1} \text{ and} \\ & n \bmod (\gamma(n-1)+1) = 1 \\ \leq 3\lfloor \frac{n}{\gamma(n-1)+1} \rfloor - 1 & \text{if } \frac{1}{2} > \gamma \geq \frac{2}{n-1} \text{ and} \\ & n \bmod (\gamma(n-1)+1) \geq 2 \\ \leq n - 1 & \text{if } \gamma = \frac{1}{n-1} \end{cases}$$

The upper bounds are realizable. In a network (e.g., a social network or a citation network) scenario, a Quasi-Clique is a set of objects that are highly interactive with each other. Therefore, a Quasi-Clique in a graph may strongly indicate the existence of a potential community. Since a Quasi-Clique contains a group of highly interacting (and thus likely highly similar in role) objects, it may be more reliable in representing relationships than individual objects. Heuristically, we can use Quasi-Clique to annotate the relationships in large scale subgraphs, which is highly desirable for solving the GER problem.

4.1.2.1 Step 1: Mining Graphs from Context through Superimposition

In order to apply the Quasi-Clique to measure contextual distances, we first need to capture this additional information of grouped entities as "graphs." Let us

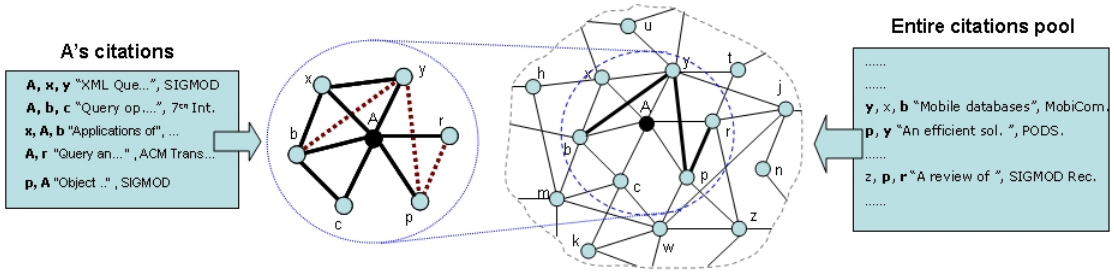


Figure 4.3: Illustration of "superimposition" of co-author data set.

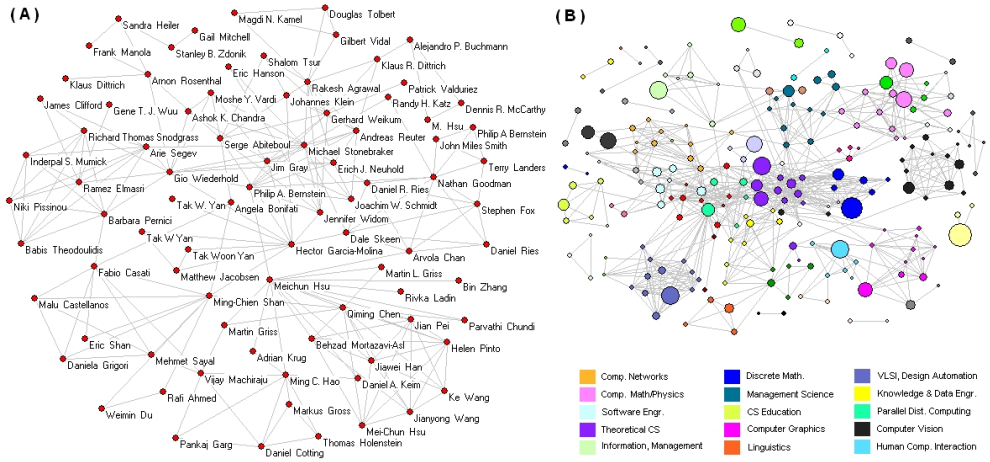


Figure 4.4: Exemplar graphs mined from the ACM data set: (a) A collaboration graph on the 2-vertex neighborhood of "Umeshwar Dayal," and (b) A venue relation graph using Jaccard similarity on the authors of venues.

informally explain this process using examples. Suppose we want to mine a graph out of an author entity A's co-author tokens: B, C, D, and E. First, a vertex is prepared for tokens, A through E, referred to as $V(A)$ through $V(E)$. Then, four co-author vertices, $V(B)$ through $V(E)$, are connected to the main vertex $V(A)$, forming a graph, g_a. Then, g_a is "superimposed" to the collaboration graph, G, that is pre-built using the entire set of co-authors from all grouped entities. For instance, if an author C had co-authored with an author D elsewhere, then now g_a will have an edge connecting $V(C)$ and $V(D)$. At the end, if all neighboring co-authors of A have co-authored each other, then g_a becomes a clique. This process is illustrated in Figure 4.3. Also, Figure 4.4(a) is a collaboration graph on partial data set of the ACM digital library. Similarly, for venue information, once we create an initial graph, g_a, we can superimpose it against some semantically

constructed graph. For instance, Figure 4.4(b) illustrates a venue graph where an edge between two venue vertices represents the "semantic" similarity of two venues (e.g., how many authors are common). If a grouped entity A has two distinct venue tokens, "VLDB" and "SIGMOD," then we first create an initial graph where A is the main vertex and there are two neighboring vertices for "VLDB" and "SIGMOD." Then, when we superimpose this initial graph against Figure 4.4(b), due to their close semantic relationship, an edge between vertices for "VLDB" and "SIGMOD" may appear. The superimposition works as long as there is a base graph (e.g., collaboration graphs, venue relation graphs) onto which an entity's graph is superimposed. For semantics-rich problem domains like citation data sets, generating such a base graph is easy.

4.1.2.2 Step 2: distQC – Using Quasi-Clique to Measure Contextual Distance

Input : A grouped entity e, an ER method M, and three parameters (α, γ, and S).

Output: k variant grouped entites, $e_v(\in E)$, such that $e_v \sim e$.

Using M, find top $\alpha \times k$ candidate entities, e_X;
$G_c(e) \leftarrow$ context graph of e;
for $e_i(\in e_X)$ **do**
  $G_c(e_i) \leftarrow$ context graph of e_i;
  $g_i \leftarrow QC(G_c(e), \gamma, S)$;
Sort $e_i(\in E_X)$ by $|g_i|$, and return top-k;

Algorithm 3: `distQC`

Once the contexts of entities are captured and represented as context graphs, their similarity can be properly modeled using *Quasi-Clique* [57]. A connected graph G is a $\gamma - quasi - complete\ graph(0 < \gamma \leq 1)$ if every vertex in the graph has degrees of at least $\gamma(|V(G)|-1)$. In a graph G, a subset of vertices $S \subseteq V(G)$ is a $\gamma - Quasi - Clique(0 < \gamma \leq 1)$ if $G(S)$ is a r-quasi-complete graph, and no proper superset of S has this property. Clearly, a 1-$Quasi-Clique$ is a clique. In a network (e.g., a social network or a citation network) scenario, a $Quasi - Clique$ is a set of objects that are highly interactive with each other. Therefore, a $Quasi - Clique$ in a graph may strongly indicate the existance of a potential community. Since

Table 4.1: Summary of data sets.

Data set	Domain	# of grouped entities	# of tuples in all entities
ACM	Computer Science	707,368	1,037,968
EconPapers	Economics	18,399	20,486
BioMed	Medical	24,098	6,169
IMDB	Entertainment	935,707	446,016

a $Quasi-Clique$ contains a group of highly interacting (and thus likely highly similar in role) objects, it may be more reliable in representing relationships than individual objects.

While γ value indicates the compactness of $Quasi-Clique$, another parameter that is of interest is the number of vertices of $Quasi-Clique$. We denote this parameter as S. For a graph G, therefore, functions: (1) QC(G, γ, S) returns a $\gamma-Quasi-Clique$ graph g from G with $|V(g)| \geq S$ if it exists, and (2) QC(G_1, G_2, γ, S) returns a *common* $\gamma-Quasi-Clique$ graph g of G_1 and G_2 with $|V(g)| \geq S$. Then, $|g|$ indicates how strongly two graphs G_1 and G_2 are related, and can be used as a "distance." Our two-step GER algorithm, distQC, is shown in Algorithm 3. Given an entity $e(\in E)$, to locate matching k variant entities, the distQC algorithm first relies on any existing ER solutions, and selects α (e.g., $2 \leq \alpha \leq 3$) times more number of candidates than k as an extra. Since we try to improve precisions by reducing false positives, once we get more candidates, if we can boost up those correct variants up into higher ranks in subsequent steps, then our aim can be met. γ value controls the "quasiness" of the graph, and S works as the minimum filter.

4.1.3 Experimental Set-up

To validate our proposal, we first gathered four real data sets from diverse domains – ACM, BioMed, EconPapers, and IMDB – as shown in Table 4.1. Ideally, in order to validate our proposal to the GER problem, we need canonical and variant entities (i.e., solution set). However, in general, manually gathering such a solution set is a labor-intensive task. Therefore, we prepared only one "real" solution set against ACM data set, and four synthetically generated artificial solution sets – a total of five test sets. Since we have solutions for all five test sets, we can check the correctness at the end.

Real Case. From the ACM data set, we have gathered 47 real cases of canonical and variant grouped entities. Each case has one designated canonical entity and on average two variant entities. Furthermore, each grouped entity has on average about 24.6 tuples in it (i.e., each author has 24.6 citations). Real cases include variants as simple as "Luis Gravano" vs. "L. Gravano" (i.e., abbreviation) and "Alon Levy" vs. "Alon Halevy" (i.e., last name change) to as challenging as "Shlomo Argamon" vs. "Sean Engelson" (i.e., little similarity) or even ten variants of "Jeffrey D. Ullman" of Figure 1.1. Since there are a total of 707,368 entities in the ACM data set, locating all variants correctly without yielding any false positives is a challenging task.

Synthetic Case. We synthetically generated a test case for each data set as follows. For each data set, we first pick 100 grouped entities with enough tuples in them. For instance, a canonical entity "Jeffrey D. Ullman" in the ACM data set has 266 tuples (i.e., citations) and 114 co-authors. Then, we made up a variant entity by either abbreviating the first name ("J. D. Ullman") or injecting invalid characters to the last name ("J. D. X-Ullman-Y") in 7:3 ratio. Then, Both canonical and variant entities carry halves, 133, of the original tuples. The goal is then to identify the artificially generated variant out of the entire entities – e.g., 707,368 in ACM and 935,707 in IMDB. Note that we could have generated variants with more difficult names using, for instance, alternation ("Ullman, Jeffrey D."), contraction ("JeffreyD. Ullman"), omission ("Jeffrey Ullman"), or any combinations of these. However, since we apply regular or Quasi-Clique based distance functions to the contents of the grouped entities (i.e., citations or movie lists), not to the names, such a variation attributes little to the results. Furthermore, due to the large number of entities in data sets, the first step of the algorithm often yields thousands of entities, out of which our Quasi-Clique-based method needs to find 1 or 2 variants in the second step – non-trivial task already.

Evaluation Metrics. Suppose there are R hidden variants. After top-k candidate variants are returned by an algorithm, furthermore, suppose that only r candidates are correct variants and the remaining $k - r$ candidates are false positives. Then, $\mathit{Recall} = \frac{r}{R}$ and $\mathit{Precision} = \frac{r}{k}$. Since this traditional recall and precision do not account for the quality of ranking the right variants in the answer window

k, the *Ranked Precision* measures precision at different cut-off points [36]. For instance, if the topmost candidate is a correct variant while the second one is a false positive, then we have 100% precision at a cut-off of 1 but 50% precision at a cut-off of 2. In our context, we set the cut-off points dynamically to be all ranks at which only "correct" variant is found, referred to as C. Formally, *Ranked Precision* $= \frac{\sum_{i \in C} precision_i}{k}$, where $precision_i$ is the precision at a cut-off of i. Finally, the *Average Recall (AR)* and *Average Ranked Precision (ARP)* are the averages of recall and ranked precision over all cases (i.e., 47 cases for real ACM test set and 100 cases for each of the four synthetic test sets).

Compared Methods. Recall that the goal of the proposed two-step algorithm is to avoid false positives, and thus improve overall precision and recall. To see the effectiveness of our distQC, we conducted various traditional distance metrics in the first step to get initial candidate variants. In the second step, we applied the distQC-based metric to the candidate variants to get the final answers. Then, we compared the performance of "before" and "after" distQC-based metric was applied. In the first step, we experimented the following distance metrics: *Monge-Elkan (ME)*, *Jaro (JR)*, *Jaccard (JC)*, *TF/IDF (TI)*, *Jensen-Shannon (JS)*, *Felligi-Sunter (FS)*, *SoftTFIDF (ST)* and *Cosine similarity (CS)*. For the details, please refer to [65]. In addition, we also tried one of the state-of-the-art algorithms to the GER problem – *IntelliClean (IC)* [46]. The IntelliClean solves the GER problem by (1) selecting a few important context attributes by the associative rule mining; (2) building a concept hierarchy on the attributes (e.g., values like "VLDB" and "SIGMOD" for venue attribute are clustered under "Database", which in turn falls under "Systems" etc.); and (3) measuring distances to identify variants. Authors reported good recall and precision using a manually-generated concept hierarchy in [46]. Since our test sets are too large to manually generate such a concept hierarchy, we extended the IntelliClean as follows: First, we measure all pair-wise distances among venues using Jaccard similarity, $\frac{|A \cap B|}{|A \cup B|}$, where A and B are author sets of venues X and Y. The result is shown in Figure 4.4(b). Once we can get an $n \times n$ matrix M, where $M[i][j]$ tells the distance between venues i and j, venues can be grouped into a concept hierarchy using popular clustering methods such as hierarchical clustering or k-means clustering.

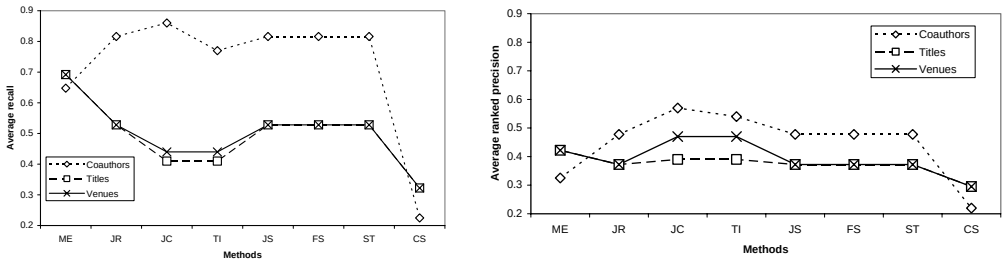


Figure 4.5: Results of eight distance metrics against ACM real case.

4.1.4 Experimental Results

1. Results of existing distance functions. Our proposed method, distQC, first uses one of the existing distance functions in the first step of distQC. Therefore, first, we attempted to identify distance functions that work best for the given GER problem. In this setting, the distance of two entities, $dist(e_c, e_v)$, is measured as that of the contents (as tokens) of two entities, $dist(tokenlist(e_c), tokenlist(e_v))$, after stop words such as "the" are pre-processed first. Figure 4.5 illustrates performance results of eight distance functions against the ACM real case. Results for other data sets show similar pattern and thus omitted. In general, token-based distance functions such as Jaccard similarity (JC) and TF/IDF (TI) show a good combination of Average Recall (AR) and Average Ranked Precision (ARP). Furthermore, for the real ACM data set, co-author information proves to be the most effective attribute. Comparing tokens of venues and titles to measure distances yield a rather poor AR and ARP due to noisy data. For instance, the venue attribute often has the same conference in diverse names (e.g., "JCDL" vs. "Joint Conference on Digital Libraries"). Therefore, functions using textual similarities do not necessarily work well. In particular, note that the low average ranked precision for all distance functions that we examined due to a large number of false positives – a good motivation for our Quasi-Clique technique later. In the proposed two-step approach, since both JC and TI perform the best in Figure 4.5, we use them as the distance metrics in the first step of distQC.

2. Parameter tuning test for distQC. distQC has two parameters to set: (1) γ for the compactness of Quasi-Clique; and (2) $\mathcal{S}$ for the minimum size of Quasi-Clique. In this test, we attempt to see if the performance of distQC is sensitive to either parameters of Quasi-Clique. Figure 4.6 illustrates AR and ARP

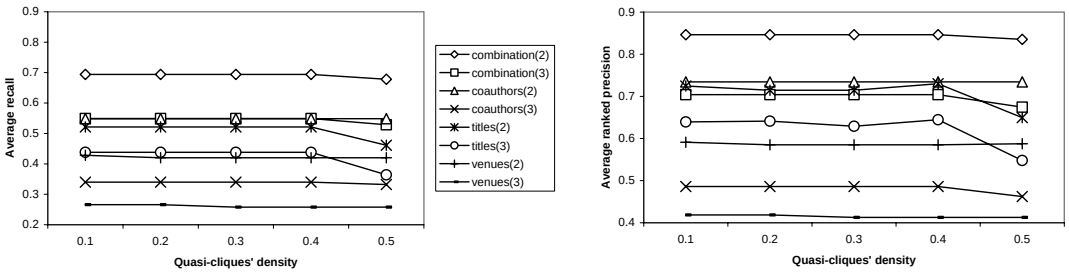


Figure 4.6: Parameter tuning test for distQC (e.g., "coauthors(2)" indicates that distQC is performed on co-authors with the minimum size 2.)

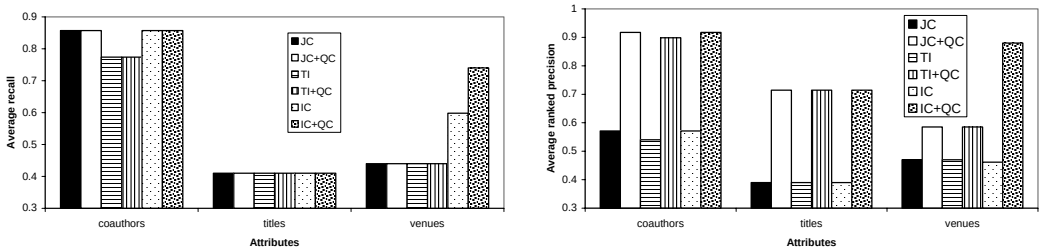
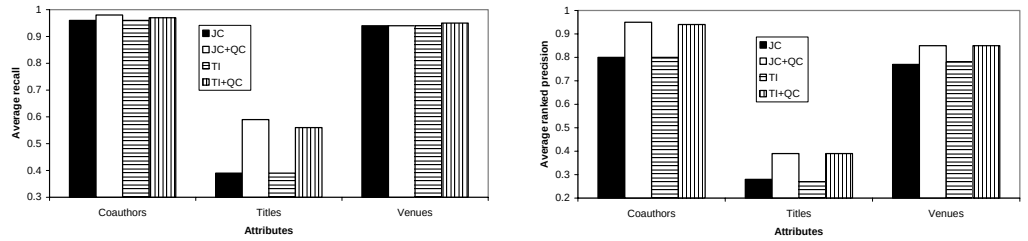


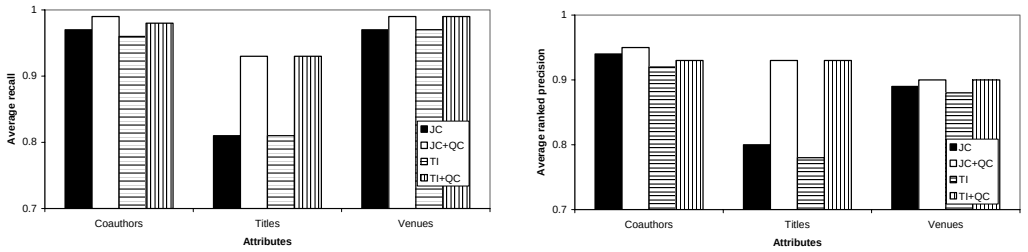
Figure 4.7: Real test case for the ACM data set.

of Quasi-Clique using the ACM real case. In every graph, each object in a group is connected to at least a portion γ ($0 < \gamma \leq 1$) of the other objects in the same group. As γ is increased from 0.1 to 0.5, and the minimum size of distQC, $\mathcal{S}$, is set to either 2 or 3. Note that Quasi-Clique perform consistently regardless of either parameters. Therefore, we set $\gamma = 0.3$ and $\mathcal{S}$=3 in the remaining test.

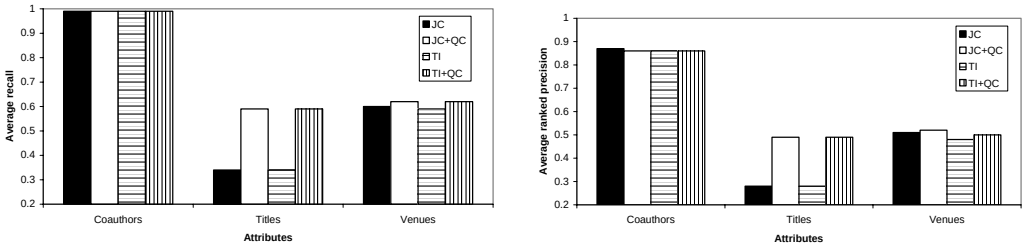
3. Quasi-Clique on the ACM real case. Next, we experiment to see if Quasi-Clique can improve the performance of one-step distance functions. We first measure the performance of three distance metrics JC, TI, and IC. Then, to each, Quasi-Clique is applied as the second step of distQC, and the performance is measured as JC+QC, TI+QC, and IC+QC. In particular, we vary two clustering algorithms to generate a concept hierarchy for IC – hierarchical clustering and k-means clustering (not shown) – both show similar performance. Figure 4.7 illustrates AR and ARP of these six methods. Note that Quasi-Clique improved ARP visibly. For instance, the ARP of JC+QC (resp. TI+QC) significantly improves from JC (resp. TI) on co-authors. On average, ARP improves by 63%, 83%, and 46% for three attributes, respectively. In general, JC and TI are simple distance metrics, measuring distances based on the occurrences of common tokens. Since some authors have name initials and common first names on co-author data,



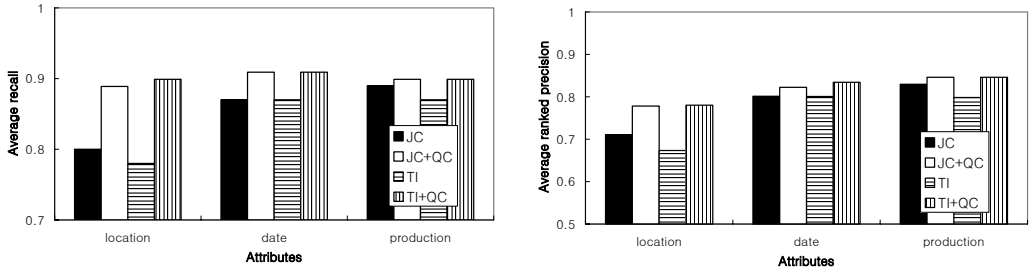
(a) ACM



(b) EconPapers



(c) BioMed



(d) IMDB

Figure 4.8: Synthetic test cases for four data sets.

therefore, these metrics tend to generate a large number of false positives as shown in Figure 4.7. Since Quasi-Clique uses additional information as to how closely co-authors of the authors are correlated each other on graph representation, it overcomes the limitations of the simple metrics.

4. distQC on Synthetic cases. Finally, we experiment to see how effective

Quasi-Clique is against large-scale data sets with synthetically generated solution sets. For this purpose, we prepared four data sets with diverse domains – ACM, EconPapers, BioMed, and IMDB. Figure 4.8 illustrates the final results of AR and ARP for all data sets. Regardless of the distance function used in the first step of distQC (either JC or TI), the type of attributes (co-authors, titles or venues), or the type of data sets, distQC consistently improved the ARP up to 75% (BioMed/the title case). Figure 4.8(d) illustrates the performance result of IMDB – a movie data set with attributes like movie IDs, countries, distributors, editors, genres, keywords, languages, locations, producers, production companies, and release dates. An example tuple looks like (935720, "USA", "Fox Network", "Pier", "Drama", "Boat", "English", "Colusa California USA", "Katir Tirnagi", "Columbia pictures television", "December 2001"). Then, the GER problem in this domain is that given two actors, X and Y (i.e., a movie star "Jim Carey" and his name variant "J. Carey"), identify if they are the same actor or not, using their movie record list. Note that, unlike citation data sets, both correlations of inter-records and inter-attribute values are weak. For instance, authors in citations usually work with particular co-authors so that by mining hidden contextual distance between co-authors, Quasi-Clique contributes to avoiding false positives. However, not all actors work with particular editors or distributors for many movies. At the end, we select three attributes, "locations", "production companies", and "release dates", as these have more contextual information than other attributes (e.g., French actors tend to star french movies more often). Figure 4.8(d) illustrates AR and ARP of JC and TI, and their two-step correspondents, JC+QC and TI+QC. Overall, the location attribute shows relatively lower AR and ARP than release dates and production companies, in all of the methods. This is because there are a lot of uncommon tokens such as the city "Victorville" in the location attribute. Nevertheless, distQC improve the ARP by 13%, suggesting its effectiveness. Both AR and ARP of TI are low, compared to those of JC. This is because there are common attribute values, such as "USA" in locations, and "pictures" or "movies" in production companies. In TI, for weighting tokens, common tokens (e.g., "movies") would have lower weight via IDF, negatively affecting matching process. In AR, distQC show much higher than both JC and TI. Compared with citation data sets, note that our distQC algorithm performs slightly better than

string distance metrics for the IMDB data set. This is because (1) records and attribute values of an actor and his/her variant entities have no strong relationships unlike those of citations; (2) attribute values of citations are long while those of the IMDB data set are short, carrying less meaningful information; (3) many attributes in the IMDB data set contain empty values and noises.

4.1.5 Summary

Toward the grouped-entity resolution problem, we present a graph partition based approach using *Quasi-Clique*. Unlike string distance or vector-based cosine metric, our approach examines the relationship hidden under the grouped-entities. Experimental results verify that our proposed approach improves precision up to 83% at best (in Figure 4.7), but never worsens it. Therefore, as a complementary approach to existing ER solutions, we believe that our proposal can help improve the performance of conventional approaches that are only based on syntactic similarities.

4.2 Group Linkage Measure

The quality of data residing in databases gets degraded due to a multitude of reasons. Such reasons include transcription errors (e.g., lexicographical errors, character transpositions), lack of standards for recording database fields (e.g., person names, addresses), and various errors introduced by poor database design (e.g., update anomalies, missing key constraints). Data of poor quality can result in significant impediments to popular business practices: sending products or bills to incorrect addresses, inability to locate customer records during service calls, inability to correlate customers across multiple services, etc.

To be able to query and integrate such data in the presence of data quality errors, a central problem is the ability to identify whether two entities are approximately the same. When each entity is represented as a relational record, this problem is referred to as the *record linkage problem*. Depending on the type of data under consideration, various similarity measures (approximate match predicates) have been defined to quantify the closeness of a pair of records in a way that common mistakes are captured. Given any specific similarity measure, a key

algorithmic problem in this context is the *approximate match problem*: given a large relation of data records and a single query record, identify all data records whose similarity with the query record exceeds a specified similarity threshold.

Often entities are represented as groups of relational records (sharing a group ID), rather than individual relational records. For example, a household in a census survey consists of a group of persons, where each person is represented by a relational record. Similarly, an author in a digital bibliography consists of a group of publications, where each publication is represented by a relational record. In such cases, there is often a need to identify whether two entities (represented by groups of relational records) are approximately the same. For example, re-sampling strategies used in statistical census surveys would need to identify matching households between the original sampled set and the re-sampled set. Similarly, when integrating different digital bibliographies (e.g., DBLP and ACM Digital Library), there is a need to identify an author in one that matches an author in the other. We refer to the problem of determining if two entities represented as groups are approximately the same as the **group linkage** problem.

Determining if two entities, represented as groups, can be linked to each other typically requires (approximately) matching elements (each of which is a relational record) across the two groups using record linkage techniques. Intuitively, two groups can be linked to each other if:

- There is high enough similarity between "matching" pairs of individual records that constitute the two groups; the matching pairs of records do not need to be identical.

- A large fraction of records in the two groups form matching record pairs; not all records in each of the two groups need to match.

Our first technical contribution in the group linkage problem is to formalize this intuition and propose a group linkage measure based on bipartite graph matching, referred to as $BM_{sim,\rho}$, where ρ is a lower bound on the similarity between pairs of records in the two groups using record-level similarity measure sim. This measure is a natural generalization of the popular Jaccard similarity measure between two sets. In particular, if we constrain records from the two groups to match only when

they are identical, our group linkage measure reduces to the Jaccard similarity measure.

Given the $BM_{sim,\rho}$ group linkage measure, a key algorithmic problem in this context is the *approximate match* problem: given a large relation of records (each associated with a group ID) D and a single query group of records g, identify all groups of records $g_i \in D$ such that $BM_{sim,\rho}(g, g_i) \geq \theta$, where $0 \leq \theta \leq 1$ is a group similarity threshold. Computing the group linkage measure $BM_{sim,\rho}$ requires the computation of maximum weight bipartite matching, which is an expensive operation. Efficiently finding groups $g_i \in D$ with a high group linkage similarity to an input query group requires quickly eliminating the many groups that are unlikely to be desired matches.

To enable this task, our second contribution is the development of simpler group similarity measures that can be used either as lower or upper bounds to $BM_{sim,\rho}$, or as a blocking technique during a fast pre-processing step. We show that these simpler group similarity measures can be easily instantiated using SQL, permitting our techniques to be implemented inside the database system itself.

Our third contribution is a detailed experimental study validating the utility of our measures and techniques using a variety of real and synthetic data sets. We show that our measures can be implemented efficiently, are more robust and can achieve better recall than competing approaches.

4.2.1 Group Linkage: Problem

We use D to denote a relation of multi-attribute records, one of whose attributes is a group ID; r (possibly with subscripts) to denote records in D; and g (possibly with subscripts) to denote groups of records in D, i.e., a set of records that share the same value of the group ID attribute. Let *sim* denote an arbitrary record-level similarity measure, such that $sim(r_i, r_j) \in [0..1]$, where 1 denotes a perfect match. Table 4.2 summarizes the notations used throughout this chapter.

Group Linkage Measure. Consider two groups of records $g_1 = \{r_{11}, r_{12}, \ldots, r_{1m_1}\}$ and $g_2 = \{r_{21}, r_{22}, \ldots, r_{2m_2}\}$. The *group linkage measure* $BM_{sim,\rho}$ is the normalized weight of the maximum weight matching M in the bipartite graph (N =

Notation	Description
D	relation of multi-attribute records
g_1, g_2, ...	groups of records in D
r_1, r_2, ...	records in D
$sim(r_i, r_j)$	arbitrary record-level similarity function
θ	group-level similarity threshold
ρ	record-level similarity threshold
M	maximum weight bipartite matching
BM	bipartite matching based group linkage
UB, LB	upper and lower bound of BM
MAX	$max()$ based heuristic group linkage

Table 4.2: Summary of notations.

$g_1 \cup g_2, E = g_1 \times g_2$), defined as:

$$BM_{sim,\rho}(g_1, g_2) \quad = \quad \frac{\Sigma_{(r_{1i}, r_{2j}) \in M}(sim(r_{1i}, r_{2j}))}{m_1 + m_2 - |M|}$$

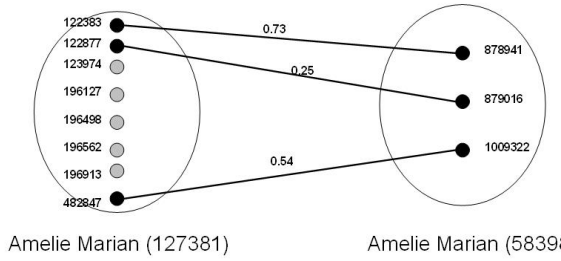
such that each $sim(r_{1i}, r_{2j}) \geq \rho$.

The numerator of $BM_{sim,\rho}$ is the weight of the maximum weight bipartite matching M using only edges with record-level similarity no less than ρ. The denominator adds up the number of edges in the matching M and the number of "unmatched" elements (i.e., records) in each of g_1 (i.e., $m_1 - |M|$) and g_2 (i.e., $m_2 - |M|$). $BM_{sim,\rho}$ is guaranteed to be between 0 and 1. When the numerator is large, it captures the intuition that there is high enough similarity between "matching" pairs of individual records that constitute the two groups; the matching pairs of records do not need to be identical. When the denominator is small, it captures the intuition that a large fraction of records in the two groups form matching record pairs; not all records in each of the two groups need to match.

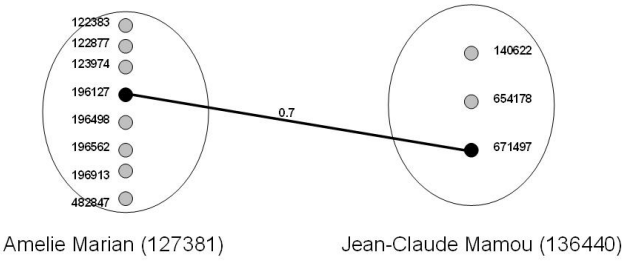
The group linkage measure $BM_{sim,\rho}$ is a natural generalization of the popular Jaccard similarity measure between two sets s_1 and s_2, defined as $\frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$. In particular, if we constrain the records from the two groups to match only when they are identical, our group linkage measure results in the same similarity value as the Jaccard similarity measure. Likewise, since $BM_{sim,\rho}$ is a generalization of the Jaccard measure, it can identify matching groups when the Jaccard fails as shown in the following example.

ID	Group name	Description
g_{127381}	Amelie Marian	canonical group
g_{583980}	Amelie Marian	matching group
g_{136440}	Jean-Claude Mamou	false positive
g_{32238}	Brendan Hills	false positive

Table 4.3: Example of "Amelie Marian".



(a) Maximum weight bipartite matching for g_{583980}.



(b) Maximum weight bipartite matching for g_{136440}.

Figure 4.9: Illustration of $BM_{cosine,0.1}$.

Consider a real example in Table 4.3 derived from the data set $R1_{DB}$ in Section 4.2.3. The goal is to match g_{127381} with g_{583980}, without being confused with false positives, g_{136440} and g_{32238}. Note that these false positives share many common tokens with g_{127381}. For instance, g_{136440} ("Jean-Claude Mamou") shares common co-author tokens of {`brendan`, `bruno`, `cassio`, `dos`, `hills`, `jean-claude`, `marian`, `mignet`, `sophie`, `tova`, `vercoustre`, `abiteboul`, `aguilera`, `amann`, `anne-marie`, `bernd`, `cluet`, `hubert`, `laurent`, `mamou`, `milo`, `santos`, `serge`, `souza`, `tessier`, `vincent`} and title tokens of {`changes`, `documents`, `evaluating`, `optimizing`} with g_{127381} ("Amelie Marian").

On the other hand, matching group g_{583980} ("Amelie Marian") shares common co-author tokens of {`gravano`, `luis`} and title tokens of {`active`, `demonstration`,

Rank	g_1	g_2	$BM_{cosine,0.1}(g_1, g_2)$
1	g_{127381}	g_{583980}	0.19
2	g_{127381}	g_{32238}	0.07
3	g_{127381}	g_{136440}	0.07

Table 4.4: Result of $BM_{cosine,0.1}$.

`repository`, `views`, `xml`, `queries`, `databases`, `detecting`, `multimedia`, `repositories`, `selection`, `xml`} with g_{127381} ("Amelie Marian"). When the Jaccard measure is applied, it would return g_{136440} ("Jean-Claude Mamou") as the most similar group to g_{127381} ("Amelie Marian") with the similarity of 0.22. However, the real matching group g_{583980} only comes second with the similarity of 0.2. On the other hand, as illustrated in Figure 4.9, when $BM_{cosine,0.1}$ is used (using *cosine* similarity as the record-level similarity measure with 0.1 as threshold), it is able to identify the correct matching group, partly due to the fact that the two matching groups share three similar record-level pairs. The result is shown in Table 4.4.

Note that when $\rho = 0$, the group linkage measure permits any pair of records $(r_{1i}, r_{2j}) \in E$ to be part of the maximum weight matching M. In practice, however, for two records to be considered approximately the same their similarity needs to exceed some threshold $\rho > 0$ using a record-level similarity function sim, both of which are parameters to the group linkage measure BM.

The main problem addressed in the group linkage problem is the *approximate match problem*. Formally, our problem can be defined as follows.

> Given a large relation of records (each associated with a group ID) D and a single query group of records g, identify all groups of records $g_i \in D$ such that $BM_{sim,\rho}(g, g_i) \geq \theta$, where $0 \leq \theta \leq 1$ is a group similarity threshold.

A related problem is the top-k version of the problem where we are interested in identifying the groups of records $g_i \in D$ whose $BM_{sim,\rho}(g, g_i)$ values are the k highest among all groups in D.

Addressing these problems are essential to be able to query and integrate entities (such as households in census surveys and authors in digital bibliographies) that are represented as groups of multi-attribute records, in the presence of data quality errors.

4.2.2 Group Linkage: Solution

4.2.2.1 Bipartite Matching

To solve the approximate match problem requires that we first identify all pairs of records $r_i \in g, r_j \in D$ such that the record-level similarity $sim(r_i, r_j) \geq \rho$. Efficient techniques are known for identifying such record pairs for a variety of record-level similarity measures, including edit distance [28] and tf.idf cosine similarity [29]. Recently, Chaudhuri et al. [12] proposed the SSJoin operator for optimizing and efficiently evaluating a variety of record-level similarity measures inside the database.

Once we have identified all such record pairs, the next step is to identify groups g_i in D such that $BM_{sim,\rho}(g, g_i) \geq \theta$. This requires the use of maximum weight bipartite matching. Bipartite matching is a well-studied problem for which efficient algorithmic solutions are known. Guha et al. [30] presented SQL realizations of the Hungarian algorithm and of an incremental strategy referred to as SSP for identifying maximum weight perfect bipartite matchings. While reasonably efficient for identifying the maximum weight bipartite matching (and hence $BM_{sim,\rho}$) for a pair of groups, applying such a strategy for every pair of groups is infeasible when the database relation contains a very large number of groups.

Efficiently finding groups $g_i \in D$ with a high group linkage similarity to input query group g requires quickly eliminating the many groups that are unlikely to be desired matches. To enable this task, we next develop simpler group similarity measures that can be used either as bounds to $BM_{sim,\rho}$, or as a blocking technique during a fast pre-processing step.

4.2.2.2 Greedy Matching

Maximum weight bipartite matching is computationally expensive because of the requirement that no node in the bipartite graph can have more than one edge incident on it. We can quickly obtain (upper and lower) bounds on $BM_{sim,\rho}$ by relaxing this requirement, using the following greedy strategy for each pair of groups g_1, g_2.

- For each record $r_i \in g_1$, find a record $r_j \in g_2$ with the highest record-level

similarity among those with $sim() \geq \rho$. For the pair of groups g_1, g_2, let $S1$ denote the set of all such record pairs.

- Similarly, for each record $r_j \in g_2$, find a record $r_i \in g_1$ with the highest record-level similarity among those with $sim() \geq \rho$. For the pair of groups g_1, g_2, let $S2$ denote the set of all such record pairs.

Note that neither $S1$ nor $S2$ may be a matching. In $S1$, the same record in g_2 may be the target of more than one record in g_1. Similarly, in $S2$, the same record in g_1 may be the target of more than one record in g_2. However, these sets can be used to quickly obtain bounds for $BM_{sim,\rho}$, using the following formulas.

Upper and Lower Bounds. Consider two groups of records $g_1 = \{r_{11}, r_{12}, \ldots, r_{1m_1}\}$ and $g_2 = \{r_{21}, r_{22}, \ldots, r_{2m_2}\}$. The upper and lower bounds of *group linkage measure* $BM_{sim,\rho}$ is defined as:

$$\begin{aligned} UB_{sim,\rho}(g_1, g_2) &= \frac{\Sigma_{(r_{1i}, r_{2j}) \in S1 \cup S2}(sim(r_{1i}, r_{2j}))}{m_1 + m_2 - |S1 \cup S2|} \\ LB_{sim,\rho}(g_1, g_2) &= \frac{\Sigma_{(r_{1i}, r_{2j}) \in S1 \cap S2}(sim(r_{1i}, r_{2j}))}{m_1 + m_2 - |S1 \cap S2|} \end{aligned}$$

where $S1$ and $S2$ are defined above.

Comparing $BM_{sim,\rho}$ of Definition 4.1 and $UB_{sim,\rho}$ of Definition 4.2.2.2, we can infer that the numerator of $UB_{sim,\rho}$ is at least as large as the numerator of $BM_{sim,\rho}$, and the denominator of $UB_{sim,\rho}$ is no larger than the denominator of $BM_{sim,\rho}$. Similarly, by comparing $BM_{sim,\rho}$ of Definition 4.1 and $LB_{sim,\rho}$ of Definition 4.2.2.2, we can infer that the numerator of $LB_{sim,\rho}$ is no larger than the numerator of $BM_{sim,\rho}$, and the denominator of $LB_{sim,\rho}$ is at least as large as the denominator of $BM_{sim,\rho}$. As a result, we have the following result.

Consider two groups of records g_1 and g_2. Then, for this pair of groups, we have that

$$LB_{sim,\rho}(g_1, g_2) \leq BM_{sim,\rho}(g_1, g_2) \leq UB_{sim,\rho}(g_1, g_2)$$

That is, $BM_{sim,\rho}$ is bounded.

The advantage of these simpler group similarity measures $UB_{sim,\rho}$ and $LB_{sim,\rho}$ is that these bounds can be easily and efficiently instantiated using SQL, permitting them to be implemented inside the database system itself. Quickly computing

$UB_{sim,\rho}$ and $LB_{sim,\rho}$ can help us efficiently address our approximate match problem as follows.

1. Since $UB_{sim,\rho}$ is an upper bound on $BM_{sim,\rho}$, if $UB_{sim,\rho}(g_1, g_2) < \theta$, then it must be the case that $BM_{sim,\rho}(g_1, g_2) < \theta$. Hence, (g_1, g_2) is guaranteed to not be part of the answer to the approximate match problem and can be pruned away.

2. Since $LB_{sim,\rho}$ is a lower bound on $BM_{sim,\rho}$, if $LB_{sim,\rho}(g_1, g_2) \geq \theta$, then it must be the case that $BM_{sim,\rho}(g_1, g_2) \geq \theta$. Hence, (g_1, g_2) is guaranteed to be part of the answer to the approximate match problem and can be selected without computing the more expensive $BM_{sim,\rho}$.

3. Only when $LB_{sim,\rho}(g_1, g_2) < \theta \leq UB_{sim,\rho}(g_1, g_2)$, the more expensive $BM_{sim,\rho}(g_1, g_2)$ computation would be needed.

The group similarity measures $UB_{sim,\rho}$ and $LB_{sim,\rho}$ can be used in a pre-processing step to speed up computation of the answers to the approximate match problem, *without incurring any false negatives*.

4.2.2.3 Heuristic Measure

In this section, we describe a group similarity measure that is even simpler (and hence faster to compute) than $UB_{sim,\rho}$ and $LB_{sim,\rho}$, and can also be used during a pre-processing step (called *blocking*).

Heuristic Group Linkage Measure. For a pair of groups (g_1, g_2), a heuristic measure using the *max* function is defined as follows:

$$MAX_{sim,\rho}(g_1, g_2) = \max_{(r_{1i}, r_{2j}) \in g_1 \times g_2} (sim(r_{1i}, r_{2j}))$$

The key intuition behind the use of $MAX_{sim,\rho}$ is that it is quite likely in practice that pairs of groups with a high value of $BM_{sim,\rho}$ will share at least one record with a high record-level similarity. By quickly identifying groups with the largest values of $MAX_{sim,\rho}$, we can avoid computing $BM_{sim,\rho}$ on a significant number of groups. Unlike the use of $UB_{sim,\rho}$ and $LB_{sim,\rho}$ as described above, however, this is a heuristic, without any guarantees of avoiding false negatives. We experimentally evaluate the utility of $MAX_{sim,\rho}$ in Section 4.2.3.

```
insert UB select T.gid1 as gid1, T.gid2 as gid2,
   T.s/cast(T.cnt+abs(G1.numOfElm-T.cnt)+abs(G2.numOfElm-T.cnt) as float) as sim
from (
        select count(*) as cnt, sum(B.sim) as s, B.gid1 as gid1, B.gid2 as gid2
        from
        (
           select A.eid1 as eid1, max(A.sim) as sim, A.gid1 as gid1, A.gid2 as gid2
           from
           (
              select S.eid1 as eid1, S.sim as sim, E1.gid as gid1, E2.gid as gid2
              from GrpToElm1 E1, GrpToElm2 E2, ElmSim S
              where E1.eid = S.eid1 and E2.eid = S.eid2
           ) A
           where A.eid1 in (select S.eid1
                            from GrpToElm1 E1, GrpToElm2 E2, ElmSim S
                            where E1.eid = S.eid1 and E2.eid = S.eid2)
           group by A.eid1, A.gid1, A.gid2
        ) B
        group by B.gid1, B.gid2
     ) T, Group1 G1, Group2 G2
where G1.gid = T.gid1 and G2.gid = T.gid2 ...
// if there is a threshold
having T.s/cast(T.cnt+abs(G1.numOfElm-T.cnt)+abs(G2.numOfElm-T.cnt)
as float) > 0
```

Table 4.5: The SQL statement for $UB_{sim,\rho}$.

4.2.2.4 Implementation

It becomes increasingly important to efficiently instantiate data quality algorithms (e.g., record or group linkage algorithms) using SQL, enabling implementation on top of any DBMS. Recent examples of this trend include the SQL implementation of matching algorithms [29] and the merge algorithms in [30]. We built $BM_{sim,\rho}$ by extending SSP in [30], and implemented all of the $UB_{sim,\rho}$, $LB_{sim,\rho}$, and $MAX_{sim,\rho}$ in SQL.

The basic logic of $UB_{sim,\rho}$ is: for each element in g_i, find the best match in g_j. It is possible that g_i has multiple edges incident on it (i.e., match). Then, the similarity between g_i and g_j is computed as the sum of edge weights divided by (number of matched edges + number of unmatched nodes in g_i and g_j). Some of the tables being used include:

- $Group_k$(gid, name, numOfElm): records the number of elements per group.
- $GrpToElm_k$(gid, eid): associates each group with elements in it.
- $ElmSim$(eid1, eid2, sim): records, for all pair-wise records, record-level similarity ($\geq \rho$) using cosine measure with TF/IDF weighting.

```
insert MAX select T.gid1 as gid1, T.gid2 as gid2, max(T.sim) as sim
from (
      select E1.gid as gid1, S.eid1 as eid1, E2.gid as gid2, S.eid2 as eid2, S.sim as sim
      from GrpToElm1 E1, GrpToElm2 E2, ElmSim S
      where E1.eid = S.eid1 and E2.eid = S.eid2
     ) T
group by T.gid1, T.gid2
```

Table 4.6: The SQL statement for $MAX_{sim,\rho}$.

Table 4.5 and Table 4.6, for instance, shows a snippet of SQL statement to implement $UB_{sim,\rho}$ and $MAX_{sim,\rho}$.

Table 4.7 shows the SQL statement for maximum weighted bipartite matching, based on the *Successive Shortest Paths (SSP)*. Block 1 initializes the T relation to the most expensive edge out of position 1. We then iterate over all positions from 2 to k. Within each iteration, block 2 computes the transitive closure from the current position to all unmatched positions. Block 3 picks an unmatched tuple with the largest similarity from the current position and block 4 computes the path from the current position to this unmatched tuple. Finally, block 5 updates the old solution using this path to obtain the new solution which includes the current position.

The similarity measure between two elements can be estimated using TF/IDF Cosine similarity, as illustrated in Table 4.8. Specifically, we create the following relations for TF/IDF Cosine similarity (Let's assume that there are two relations i and j, and there are $k \in n$ attributes).

- Token$i$(element-id, token): a tuple is associated with an occurrence of *token* in Relation i tuple with element-id.
- IDFi(token, idf): a tuple indicates that *token* has inverse document frequency idf_{token} in Relation i.
- TFi(element-id, token, tf): a tuple indicates that *token* has term frequency tf_{token} for Relation i tuple with element-id.
- Length$i$(element-id, length): a tuple indicates that the weight vector associated with Relation i tuple with element-id has a Euclidean norm *length*.
- ElmToToken$i$(element-id, token, weight): a tuple indicates that *token* has normalized *weight* in Relation i tuple with element-id.

```
delete from T insert into T(elem, pos) select max(Graph.elem), pos
from Graph where pos = 1 and cost = (select max(cost) from Graph
where pos = 1) group by pos;

declare @N int; set @N = (select count(elem) from Graph); declare @K
int; set @K = (select count(pos) from Graph); declare @INDEX int;
set @INDEX = 2; while (@INDEX <= @K)
  begin
    delete from R1
    insert into R1(selem, delem, cost)
    select T.elem, G2.elem, G2.cost-G1.cost from T, Graph G1, Graph G2
    where T.elem = G1.elem and T.pos = G1.pos and T.pos = G2.pos;
    delete from Reach4
    insert into Reach4(elem, parent, cost)
    select elem, @N+1, cost from Graph where Graph.pos = @INDEX;
    declare @ITER int
    set @ITER = 1
    while (@ITER <= @INDEX)
      begin
        delete from RR
        insert into RR(elem, parent, cost)
        select Y.elem, X.elem, X.cost+R1.cost from Reach4 X, R1, Reach4 Y
        where X.elem = R1.selem and Y.elem = R1.delem and X.cost+R1.cost < Y.cost;
        delete from R41
        insert into R41(elem, cost)
        select elem, max(cost) from RR group by elem;
        delete from Reach4 where elem in (select elem from R41);
        insert into Reach4 (elem, parent, cost)
        select RR.elem, max(RR.parent), RR.cost from RR, R41
        where RR.elem = R41.elem and RR.cost = R41.cost group by RR.elem, RR.cost;
        set @ITER = @ITER + 1;
      end

    delete from Augment
    insert into Augment
    select top 1 elem from Reach4 where elem not in (select elem from T) order by cost;

    with AugmentingPath(elem, parent, lv) as
    (
       select Reach4.elem, Reach4.parent, 0 from Reach4, Augment
       where Reach4.elem = Augment.elem
         union all
       select Reach4.elem, Reach4.parent, AugmentingPath.lv+1 from Reach4, AugmentingPath
       where Reach4.elem = AugmentingPath.parent and AugmentingPath.lv < @N
     )
     delete from AugmentingPath1
     insert into AugmentingPath1
     select elem, parent from AugmentingPath;

     insert into T values (@N+1, @INDEX);
     delete from TempT
     insert into TempT
     select AugmentingPath1.elem, T.pos from T, AugmentingPath1
     where T.elem = AugmentingPath1.parent;
     delete from T where elem in (SELECT parent FROM AugmentingPath1);
     insert into T SELECT * from TempT;

     set @INDEX = @INDEX + 1;
   end
```

Table 4.7: The SQL statement for maximum weighted bipartite matching.

- Attr$k$ElmSim(element-id, element-id, sim): a tuple indicates the similarity between element-id in Relation i and element-id in Relation j in Attribute k.

- ElmSim(element-id, element-id, sim): a combination of n AttrElmSim relations.

Then, top-k entity names selected by the highest similarity of their elements are grouped into the same block. For example, suppose two author entities "J. D. Ullman" (a canonical author) and "Jeff. Ullmann" (a name variant), where "J. D. Ullman" has an element e_1: ([Hopcroft, Aho],[Schema, Matching],[JCDL]), and "Jeff. Ullmann" has two elements e_2: ([Aho,Maier],[Value, Matching],[VLDB]) and e_3: ([Aho],[Semantic,Matching],[JCDL]). The TF/IDF Cosine similarities ($sim(e_i, e_j)$) between two elements e_i and e_j are computed as follows (attribute weights $\alpha + \beta + \gamma = 1$ – since the co-author and venue attributes tend to contain "noise" data (e.g., "A."/"E." in co-authors or "Int'l"/"Conf." in venues), we give higher weights to the title attribute than to co-authors and venues.):

- $sim(e_1, e_2) = \alpha\ sim(co{-}author(e_1), co{-}author(e_2)) + \beta\ sim(title(e_1), title(e_2)) + \gamma\ sim(venue(e_1), venue(e_2))$

- $sim(e_1, e_3) = \alpha\ sim(co{-}author(e_1), co{-}author(e_3)) + \beta\ sim(title(e_1), title(e_3)) + \gamma\ sim(venue(e_1), venue(e_3))$

Meanwhile, we can implement the SQL statement for Jaccard distance in Table 4.9

4.2.3 Experimental Set-up

We have introduced two versions of the group linkage problem – threshold and top-k. In the experiments, we used the top-k version: given a group g_i and the answer window k, return the top-k matching groups to g_i. We choose to do so in order to be able to evaluate both *performance* as well as *quality* of the answers.

To evaluate our group linkage proposal, we extracted various data sets, as summarized in Table 4.10, from the ACM and DBLP citation digital libraries. These libraries typically have a list of citations per author. Therefore, we can treat each author as a *group*, citations of an author as *records* in a group, and linkage between authors as the *group linkage* problem. By using author names as

```
insert into Sizei (s) select count(*) from Di;

insert into IDFi (token, idf) select T.token,
log(S.s)-log(count(distinct T.eid)) from Tokeni T, Sizei S group by
T.token, S.s

insert into TFi (eid, token, tf) select T.eid, T.token, count(*)
from Tokeni T group by T.eid, T.token

insert into Lengthi (eid, length) select T.eid,
sqrt(sum(I.idf*I.idf*T.tf*T.tf)) from IDFi I, TFi T where I.token =
T.token group by T.eid

insert into ElmToTokeni (eid, token, weight) select T.eid, T.token,
I.idf*T.tf/L.length from IDFi I, TFi T, Lengthi L where I.token =
T.token and T.eid = L.eid

insert into AttriElmSim (eid1, eid2, sim) select T1.eid, T2.eid,
sum(T1.weight*T2.weight) from ElmToTokeni T1, ElmToTokenj T2 where
T1.token = T2.token group by T1.eid, T2.eid having
sum(T1.weight*T2.weight) >= 0

insert into ElmSim (eid1, eid2) select eid1, eid2 from Attr1ElmSim
  union
select eid1, eid2 from AttrkElmSim
  union
select eid1, eid2 from AttrnElmSim

update ElmSim set sim = S.sim + T.sim from ElmSim S, Attr1ElmSim T
where T.eid1 = S.eid1 and T.eid2 = S.eid2

update ElmSim set sim = S.sim + 2*T.sim from ElmSim S, AttrkElmSim T
where T.eid1 = S.eid1 and T.eid2 = S.eid2

update ElmSim set sim = S.sim + T.sim from ElmSim S, AttrnElmSim T
where T.eid1 = S.eid1 and T.eid2 = S.eid2

update ElmSim set sim = sim / (n+1)
```

Table 4.8: The pre-processing SQL statement for computing TF/IDF cosine similarities among records of groups.

```
insert into J1 select distinct E.gid, T.token from GrpToElm1 E,
Token1 T where E.eid = T.eid and T.cat = @I

insert into J2 select distinct E.gid, T.token from GrpToElm2 E,
Token2 T where E.eid = T.eid and T.cat = @I

insert into J1CNT select J.gid, count(*) from J1 J group by J.gid

insert into J2CNT select J.gid, count(*) from J2 J group by J.gid

insert into INTER select J1.gid, J2.gid, count(*) from J1, J2 where
J1.token = J2.token group by J1.gid, J2.gid

insert into Jaccard1 select S.gid1 as gid1, S.gid2 as gid2,
R.cnt/cast((N1.cnt+N2.cnt-R.cnt) as float) from NpJGrpSim S, J1CNT
N1, J2CNT N2, INTER R where (S.gid1 = N1.gid and S.gid2 = N2.gid)
      and (S.gid1 = R.gid1 and S.gid2 = R.gid2)
order by S.gid1, S.gid2
```

Table 4.9: The SQL statement for Jaccard.

Notation	Left (all authors with at least 5 citations) Right
$R1_a$	279 authors with a keyword *XML* in titles from DBLP 700,000 authors from ACM
$R1_b$	265 authors with a keyword *Query* in titles from ACM 400,000 authors from DBLP
$R2_{DB}$	100 authors with keywords *Query* and *Schema* in titles from DBLP 700,000 authors from ACM
$R2_{AI}$	100 authors with keywords *Reasoning* and *Recognition* in titles from DBLP 700,000 authors from ACM
$R2_{Net}$	100 authors with keywords *ATM* and *TCP* in titles from DBLP 700,000 authors from ACM
$S1_a$	279 authors with a keyword *XML* in titles from DBLP 700,000 authors from ACM + 279 $\frac{1}{3}$-type dummies
$S1_b$	279 authors with a keyword *XML* in titles from DBLP 700,000 authors from ACM + 279 3-type dummies
$S2_s$	100 authors with keywords *Memory* and *Power* in titles from ACM 100 authors (same as left) + 100 authors with 30% errors
$S2_m$	100 authors with keywords *Memory* and *Power* in titles from ACM 100 authors (same as left) + 100 authors with 45% errors
$S2_l$	100 authors with keywords *Memory* and *Power* in titles from ACM 100 authors (same as left) + 100 authors with 60% errors

Table 4.10: Summary of data sets.

a key to link, we evaluate how well a method can link a group in the left data set (e.g., "Vannevar Bush" in DBLP) to matching groups in the right data set (e.g., "V. Bush" in ACM).

Real data sets. First, we prepared two kinds of real data sets, $R1$ and $R2$, each of which in turn has several variations, $R1_a$ and $R1_b$ for $R1$ and $R2_{DB}$, $R2_{AI}$, and $R2_{Net}$ for $R2$, respectively. In both $R1_a$ and $R1_b$ data sets, authors in the left are selected from DB venues such as SIGMOD, VLDB, and ICDE, and have at least 5 citations. The average number of citations in the left and right authors are 41 and 25 for $R1_a$ and 40 and 55 for $R1_b$. Therefore, these are rather *uniform* data sets.

Next, we create skewed data sets in three domains – (1) $R2_{DB}$ with authors in the left from DB venues such as SIGMOD, VLDB, and ICDE, (2) $R2_{AI}$ with authors in the left from AI venues such as AAAI, IJCAI, and ICML, and (3) $R2_{Net}$ with authors in the left from Network venues such as INFOCOM, MOBICOM, and SIGCOMM. The average numbers of citations in the left and matching right author sets are 30 and 9 for $R2_{DB}$, 31 and 10 for $R2_{AI}$, 22 and 6 for $R2_{Net}$. Therefore,

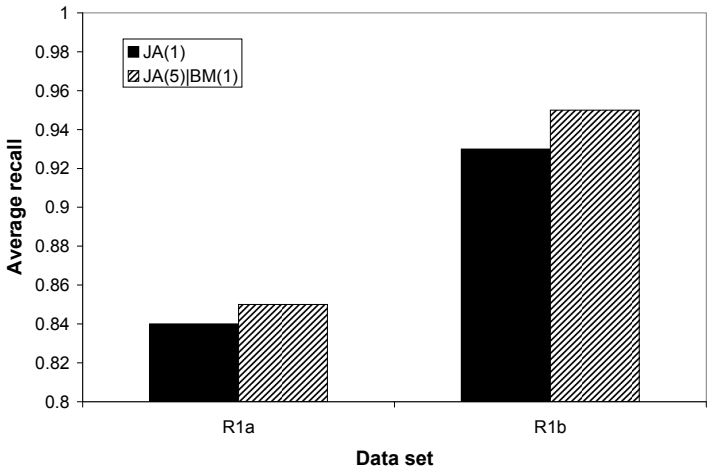


Figure 4.10: $JA(1)$ vs. $JA(5)|BM(1)$ against $R1$.

these are rather *skewed* data sets.

For all these data sets, we have manually verified whether an author in the left matches an author in the right. Note that some matched authors in the left and right have little resemblance in their names. For instance, "Shlomo Argamon" in DBLP and "Sean Engelson" in ACM are matching authors.[3] On the other hand, some authors in the left have multiple matching authors in the right. For instance, "Jeffrey D. Ullman" in DBLP has numerous matching authors in ACM such as "J. Ullman", "J. D. Ullman", and "Ullman, Jeffrey", each of which has a varying number of citations in it. In such a case, we merge all matching authors in ACM into an arbitrary one (e.g., merge "J. Ullman" and "J. D. Ullman" to "Ullman, Jeffrey"). In the end, for all data sets of Table 4.10, for a group in the left, there is only *one* matching group in the right.

Synthetic data sets. To see the effect of varying degrees of error in the data sets, we introduce controlled errors into the real data sets. First, we prepare two synthetic data sets, $S1_a$ and $S1_b$. Both are the same as the real data set $R1_a$, except that dummy authors are injected to the right. Two types of dummy authors are prepared. For an author A, its dummy author D has to use the same bag of words

[3]This is verified at his home page `http://lingcog.iit.edu/ argamon/`.

in citations as A, but the number of citations in D is either $\frac{1}{3}$ (for $S1_a$) or 3 (for $S1_b$) times the number of citations in A. Therefore, in the dummy author for $S1_a$, each citation will have many more words than a citation in A so that the number of citations becomes $\frac{1}{3}$ that of A. Similarly, in the dummy author for $S1_b$, each citation will have fewer words than a citation in A so that the number of citations becomes 3 times that of A. Note that a non-robust method such as Jaccard will get confused and return the injected dummy author from the right as the matching author of the left, incorrectly. The goal is to see if the bipartite matching group linkage method is also confused or not.

Finally, since it is not feasible to run complete bipartite matching against the 700,000 authors in the right, we generated another synthetic data set. This time, the right hand side has also a small number of authors like the left hand side so that bipartite matching can be applied without any blocking. Using the `dbgen` tool from the University of Toronto, for each author in the left, we have generated a dummy author with varying levels of errors (i.e., typos, abbreviation, omission, contraction, etc) and inserted it to the right data set. The goal is to see if a method is able to detect the correct matching author, without being confused with the dummy author.

Evaluation Metrics. For evaluating the methods, we used the average recall with an answer window size of k as well as the running time. Note that for each author a_1 in the left, there is only one other matching author a_2 in the right. Therefore, if a_2 is included in the top-k answer window for a_1, then recall becomes 1, and 0 otherwise. As the window size k increases, recall is likely to increase. At the end, we measure the *average recall* for all authors.

Compared Methods. Four methods, as summarized in Table 4.12, as well as their hybrid combinations are evaluated. Given two groups, the Jaccard (JA) method treats each group as a bag of tokens, and measures the size of intersected tokens over the size of union of tokens. When a method A is used with an answer window size of k, it is denoted as $A(k)$. When a method A with a window k_1 is used in step 1, followed by a method B with a window k_2 in step 2, the hybrid approach is denoted as $A(k_1)|B(k_2)$. For instance, the method $MAX(5)|BM(1)$ refers to the hybrid of MAX with top-5 in step 1 and BM with top-1 in step

Data set	Pre-processing time
$R2_{DB}$	1:20:05
$R2_{AI}$	1:23:57
$R2_{Net}$	1:32:48

Table 4.11: Pre-processing time for computing cosine similarity of $R2$. (hh:mm:ss).

Notation	Description
JA	Jaccard based group linkage measure
BM	$BM_{sim,\rho}$ group linkage measure
UB	$UB_{sim,\rho}$ group linkage measure
MAX	$MAX_{sim,\rho}$ group linkage measure

Table 4.12: Notation in experimentation.

2. As a record-level similarity measure, we used cosine similarity with TF/IDF weighting. This computation is done once at a pre-processing stage. The running times are shown in Table 4.11 for the case of $R2$. Other data sets took similar time, except $S2$ which took only minutes.

Summary of Set-up. Using this set-up, we seek to answer the following questions:

- Q1: Compared to the naive approach (e.g., Jaccard measure), how does $BM_{sim,\rho}$ behave? We use the $R1$, $S1$, and $S2$ data sets.
- Q2: How robust is $BM_{sim,\rho}$ in the presence of errors? We use the $S1$ and $S2$ data sets.
- Q3: How does the heuristic based $MAX_{sim,\rho}$ behave? We use the $R2$ data set.
- Q4: Can $UB_{sim,\rho}$ be used as fast pre-processing step (i.e., blocking) for $BM_{sim,\rho}$? We use the $R2$ data set.

All experiments were conducted using the Microsoft SQL Server 2005 on Pentium III 3GHZ/512MB machine.

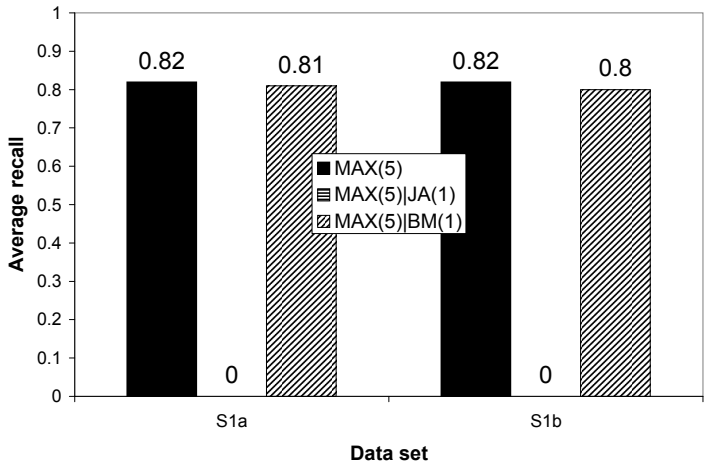


Figure 4.11: $JA(1)$ vs. $BM(1)$ against $S1$.

4.2.4 Experimental Results

1. $R1$ **real data set.** Against $R1_a$ and $R1_b$, in Figure 4.10, JA(1) (i.e., JA with window size of 1) achieved recalls of 0.84 and 0.93, respectively. Since $R1$ data sets are rather uniform with respect to the number of citations between two author groups, it is an ideal data set for JA. This explains the relatively high recall of 0.84 and 0.93. Next, we examined whether using the bipartite matching (BM) idea improves the overall recall or not. Since the graph-based BM has non-linear running time, it is not feasible to apply BM directly to $R1_a$ and $R1_b$ which has half million authors in the right. Therefore, we first apply JA(5) as the blocking method to get five candidates per canonical author, then apply BM(1) to get the top-1 matching author. Figure 4.10 shows that JA(5)|BM(1) archives recalls of 0.85 for $R1_a$ and 0.95 for $R1_b$, respectively. Note that using BM does not worsen the recall but the improvement is only minor – 1-2%. Therefore, when two groups share many common tokens and when there is no other *confusing* group with common tokens, JA works relatively well. Next, we show that this is no longer the case when JA is faced with confusing groups.

2. $S1$ **and** $S2$ **synthetic data sets.** To demonstrate that our proposed BM is more robust than JA, we created an extreme case that is the worst for JA, and see

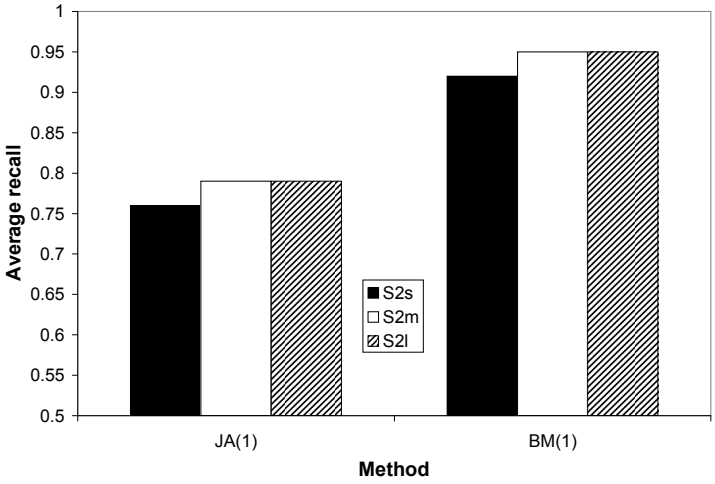


Figure 4.12: $JA(1)$ vs. $BM(1)$ against $S2$.

how BM behaves in the same environment. Note that JA determines a matching group purely based on the commonality of tokens. Therefore, if we take a canonical group, a, with N records from the left and inject a copy of it, a_c, to the right, then JA will always return a_c from the right as the matching author of a since both a and a_c have an identical bag of tokens. Based on this idea, we generate two dummy authors: (1) a_c with $\frac{N}{3}$ citations and the same bag of tokens as a ($S1_a$), and (2) a_c with $3 \times N$ citations and the same bag of tokens as a ($S1_b$).

Figure 4.11 shows the result of this comparison. To speed-up, we used MAX(5) as the *blocking* (i.e., pre-processing step), and both JA(1) and BM(1) are applied at a later step. Against the $S1_a$ data set, first, MAX(5) achieves a recall of 0.82 in the blocking stage. When both JA(1) and BM(1) are applied to the five candidate authors that MAX(5) found, MAX(5)|JA(1) has a recall of 0 while MAX(5)|BM(1) has a recall of 0.81. As per design, JA incorrectly selects dummy authors from the right, yielding 0 as recall. However, the proposed BM does not get confused and correctly identifies matching authors. Since the recall of 0.82 by MAX(5) is in a sense the ideal recall for subsequent methods, the recall of 0.81 by MAX(5)|BM(1) is a satisfactory result. The same pattern emerges for the $S1_b$ data set as well. Next, we repeated the experimentation using $S2$. This time, we did not use blocking. Instead, both JA and BM are directly applied to $S2$. Figure 4.12 shows that

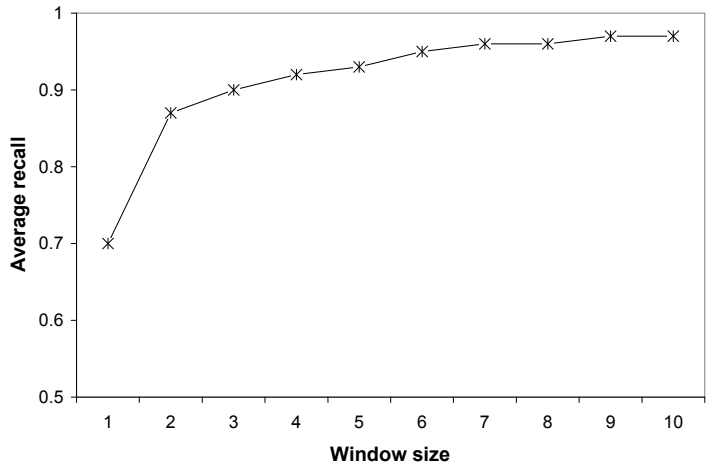


Figure 4.13: $UB(10)|BM(k)$ against $R2_{AI}$.

	$R2_{DB}$	$R2_{AI}$	$R2_{Net}$
MAX(1)	0.23	0.2	0.27
UB(1)	0.35	0.28	0.4
MAX(5)	0.24	0.2	0.26
UB(5)	0.38	0.28	0.43
MAX(10)	0.24	0.2	0.27
UB(10)	0.38	0.28	0.43
UB(5)|BM(1)	4.91	4.44	4.61
UB(10)|BM(1)	7.45	6.66	6.41

Table 4.13: Running time (per group) of $MAX(k)$, $UB(k)$, and $UB(k)|BM(1)$ against $R2$ (in sec).

$BM(1)$, regardless of the error levels, outperforms $JA(1)$ by 16-17% in recall.

Therefore, JA is a fast and simple method which can be used for group linkage, but it is prone to errors. If there exist groups with tokens similar to the canonical group, or matching groups have intolerable noises, then JA is easily confused, as shown in Figure 4.11. On the other hand, regardless of data types or error levels, BM is able to perform better group linkage.

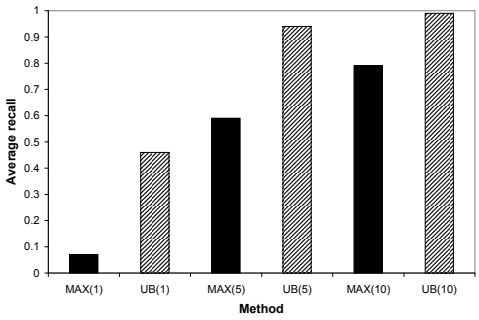
3. $R2$ **real data set.** $R2$ data sets are more challenging than $R1$ since the average number of elements per group between left and right is *skewed*. For instance,

on average, canonical authors in the left have three times more citations than corresponding authors in the right. First, Figures 4.14 (a)(c)(e) show the comparison of MAX and UB. For all three data sets, UB outperforms MAX in recall. Since MAX is heuristic based, it tends to be looser than UB (i.e., as long as there is one record pair with high similarity, group linkage will have a high similarity), allowing more errors. On the other hand, for the same reason, MAX is slightly faster than UB, as shown in Table 4.13 (first six columns).

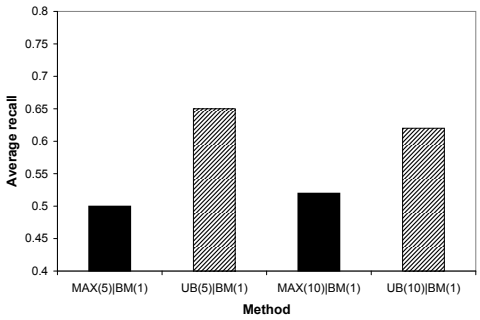
Next, Figures 4.14 (b)(d)(f) show how BM behaves when either UB or MAX is first applied as pre-processing. Regardless of data sets, UB followed by BM outperforms MAX followed by BM by 5-14%. This is consistent with Proposition 4.2.2.2, supporting our claim that UB is a good pre-processing step for BM. The answer window size in step 2 is somewhat correlated with the recall. For instance, Figure 4.13 is the case of $UB(10)|BM(k)$ against $R2_{AI}$, where $k = 1, ..., 10$ on the Y-axis.

4.2.5 Summary

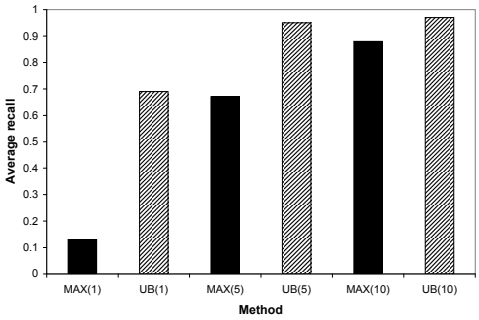
In this work, we have studied the new problem of *group linkage*, and proposed a bipartite matching based on group similarity measures, $BM_{sim,\rho}$, which naturally generalizes the Jaccard measure. In addition, we proved the upper and lower bounds of $BM_{sim,\rho}$ can be used as a filtering step for speed-up. Finally, through extensive experiments and SQL implementations, we have validated that $BM_{sim,\rho}$ is a more robust group similarity measure than others, and can efficiently detect matching groups with proper pre-processing step.



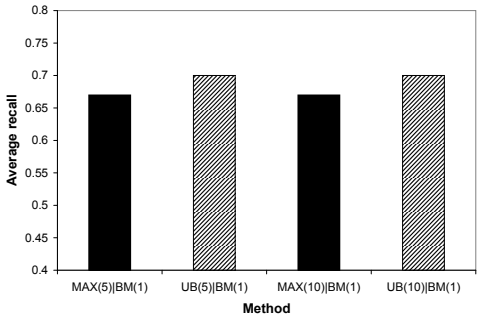
(a) $R2_{DB}$



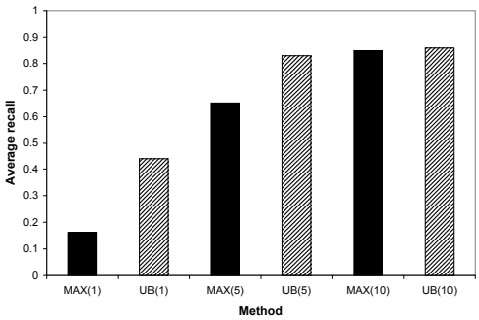
(b) $R2_{DB}$



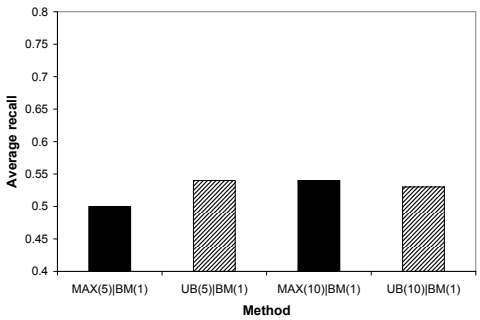
(c) $R2_{AI}$



(d) $R2_{AI}$



(e) $R2_{NET}$



(f) $R2_{NET}$

Figure 4.14: MAX vs. UB for (a)(c)(e) and $MAX|BM$ vs. $UB|BM$ for (b)(d)(f) against $R2$.

Chapter 5

The Mixed Entity Resolution (MER) Problem

In this chapter, we aim to solve an important problem that commonly occur in bibliographic DLs, which seriously degrade their data qualities. When two scholars have the same name spellings, their citation data are mistakenly merged into a single collection, leading to an incorrect citation analysis results. We call this as Mixed Entity Resolution (MER) problem. For instance, there is a collection of citation data by one of the authors, "Lin Li," in DBLP. Note that two citations by "another" scholar with the same name spelling are listed. The reason of this mixture is that there exist two computer scientists with the name "Lin Li" – one, researching energy efficient system design, at Penn State and the other, with the interest in theory of computing, at U. Nebraska, Lincoln. Thus, we investigate an effective yet scalable solution since citations in such digital libraries tend to be large-scale. After formally defining the problems and accompanying challenges, we present an effective solution that is based on the state-of-the-art sampling-based approximate join algorithm. As an alternative method, we propose name disambiguation using *multi-level graph partition*. In general, it is known that the k-way spectral clustering algorithm is the most effective method. On the other hand, as the size of a graph is significantly huge, it takes a large amount of time. To speed up such a graph partitioning algorithm but yet optimize clusters, we apply the mult-level graph partitioning algorithm to the MER problem.

5.1 Sampling-based Citation Labeling Method

As solutions to this problem, Han et al. [31] proposed two classification methods. In their *Nave Bayes*, they use Bayes theorem to classify mixed citations. In addition, they applied this problem to *Support Vector Machines* which is the best classification method. Recently Malin [51] proposed *Hierarchical clustering* to cluster mixed citations. Furthermore, Han et al. [32] proposed *K-way Spectral Clustering method*. In their method, they represent citations as matrix, and then cluster mixed citations, using Eigen values of the matrix.

However, these methods are not scalable. Since modern digital libraries tend to have a large number of citations, a scalable approach is required. Thus, we propose a *scalable citation labeling algorithm*.

In our approach, we use a sampling-based technique to quickly determine a small number of candidate authors from the entire authors in DLs.

The idea of our citation labeling algorithm is: for each citation in the collection, it tests if the citation really belongs to the given collection. First, remove an author from the citation, and then, guess back the removed author name, using additional information. If the guessed name is not equivalent to the removed name, the citation will be false citation. For details, please refer to Section 5.1.2.

5.1.1 Sampling-based Citation Labeling Method: Problem

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

Given a collection of elements (i.e, citations), C, by an entity (i.e., author), e_i, can we quickly and accurately identify false elements by another entity e_j, when e_i and e_j have the identical name spellings?

The challenge here is that since two different entities, e_i and e_j, have the "same" name spellings, one cannot easily distinguish the two entities by using similarity between their names (e.g., Jaro-Winkler). To overcome this difficulty, we propose

to exploit entity's associated information. That is, given an entity e_i, we may use additional information such as elements of the entity (i.e., his co-author list, common keywords that he often use in the titles of articles, or common publication outlets, etc).

5.1.2 Sampling-based Citation Labeling Method: Solution

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

for *each citation* c_i ($\in C$) **do**
  remove a_1 (i.e., original name) from co-author list of c_i;
  f_{cl} is applied to get a_2 (i.e., guessed name);
  if $a_1 \neq a_2$ **then**
    c_i is a false citation;
  remove c_i from C;

Algorithm 4: Citation Labeling

At the end, C has only correct citations by a_1. Therefore, if one can find a good citation labeling function f_{cl}, then one can solve the MER problem. Figure 5.1 illustrates overview of our solution.

5.1.2.1 Sampling

In general, the baseline approach has a quadratic time complexity which is prohibitively expensive for large-size DLs (e.g., the ACM digital library has about 707K authors). However, note that for a citation c, one does not need to check if c can be labeled as an author a for all authors. If one can quickly determine candidate author set from all authors (i.e., pre-filtering), then c better be tested against only the authors in candidate set. We use the following sampling technique.

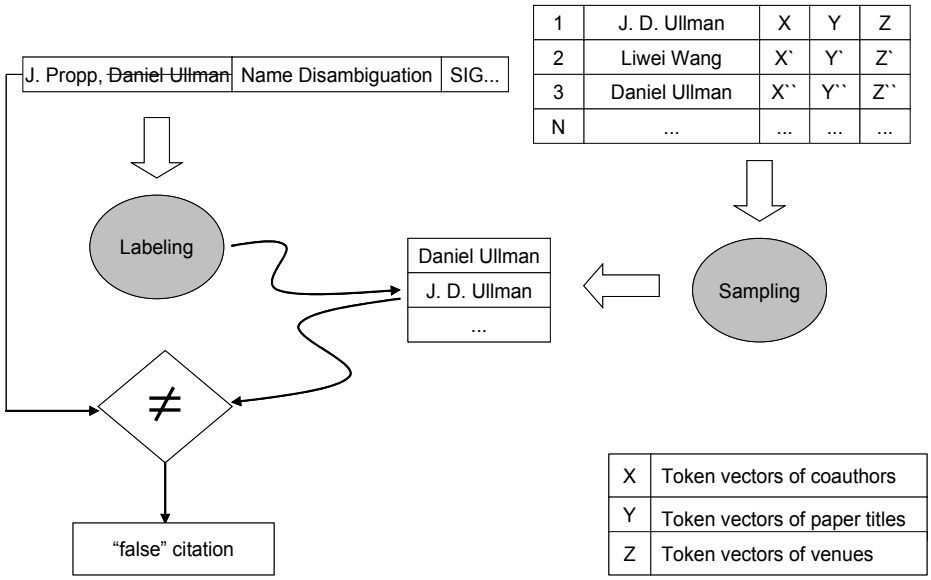


Figure 5.1: Overview of our solution to MER problem.

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

One of the state-of-the-art sampling techniques that satisfy both criteria (i.e., being fast and accurate) is the *sampling-based join approximation* method recently proposed by [29]. We adopt it to our context as follows: Their main idea is that if, for each string n_i, one is able to extract a small sample S that contains mostly strings suspected to be highly similar to n_i, then this sample S serves as a candidate set, and the remaining strings can be quickly ignored (i.e., pre-filtering). To get the "good" sample S, imagine each token from all strings has an associated weight using the TF/IDF metric in IR (i.e., common tokens in strings have lower weights while rare ones have higher weights). Then, each string t is associated with its token weight vector v_t. Suppose that, for each string t_q in a string set R_1, we want to draw a sample of size S from another string set R_2 such that the frequency C_i of string $t_i \in R_2$ can be used to approximate $sim(v_{t_q}, v_{t_i}) = \sigma_i$. That is, σ_i can be approximated by $\frac{C_i}{S} T_V(t_q)$, where $T_V(t_q) = \sum_{i=1}^{|R_2|} \sigma_i$. Then, put t_i into a candidate set only if $\frac{C_i}{S} T_V(t_q) \geq \theta$, where θ is a pre-determined threshold. This strategy assures that all pairs of strings with similarity of at least θ survive the pre-filtering stage and put into the candidate set with a desired probability, as long as the proper sample size S is given.

for *each citation* c_i $(\in C)$ **do**
  draw a sample set $S(\subseteq A)$;

Algorithm 5: Sampling

5.1.2.2 Citation Labeling Algorithm

Let us examine f_{cl} more closely. Suppose one wants to "label" a collection of citations, C, against a set of possible authors A. A naive algorithm, then, is (let ϕ be a similarity measure between a citation c_i and an author a_j):

for *each citation* c_i $(\in C)$ **do**
  examine all names $s_j(\in S)$;
  return s_j $(\in S)$ with MAX ϕ;

Algorithm 6: Scalable Citation Labeling

This approach presents a technical challenge – Since c_i and a_j are two different entities to compare in real world, the choice of good similarity measure is critical. In order to address this challenge, we propose a solution as follows:

Similarity between Citation and Author. In [31], authors reported a promising result by representing a citation as 3-tuple of co-authors, titles, and venues. Although proposed for a different problem, the idea of 3-tuple representation of citations can be adapted to our context as follows: the similarity between a citation c and an author a (hereafter, $sim(c, a)$) can be estimated as the similarity between a 3-tuple representation of c and that of a:

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

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

For instance, a citation c "E. F. Codd: A Relational Model of Data for Large

Shared Data Banks. Commun. ACM 13(6): 377-387 (1970)" is represented as: $\vec{c_c}$ = ["E.F. Codd"], $\vec{c_t}$ = ["Relational", "Model", "Data", "Large", "Shared", "Data", "Banks"], and $\vec{c_v}$ = ["Commun.", "ACM"])[1]. Similarly, an author "John Doe" with two citations ("John Doe, John Smith: Data Quality Algorithm, IQIS, 2005", and "Dario Maio, John Doe, Mark Levene: Data Cleaning for XML, ACM/IEEE Joint C. on Digital Libraries, 2005") is represented as: $\vec{a_c}$=["John Doe", "John Smith", "Dario Maio", "Mark Levene"], $\vec{a_t}$=["Data", "Quality", "Algorithm", "Cleaning", "XML"], and $\vec{a_v}$=["IQIS", "ACM/IEEE", "Joint", "C.", "Digital", "Libraries"]. In Section 5.1.3, we study the variance of handling duplicate tokens (in set and bag models). Then, the similarity of the citation c and an author "John Doe" is equivalent to: $sim(c, a)$. That is, if $sim(c, a)$ is beyond some threshold, we "guess" that c is a false citation and should have been labeled under "John Doe", not "E. F. Codd" (false positive case). When there are many such authors, we label c as the author with the maximum $sim(c, a)$.

5.1.3 Experimental Set-up

We have gathered real citation data from four different domains, as summarized in Table 3.2. Compared to previous work, all of the four data sets are substantially "large-scale" (e.g., DBLP has 360K authors and 560K citations in it). Different disciplines appear to have slightly different citation policies and conventions. For instance, Physics and Medical communities seem to have more number of co-authors per article than Economics community. Furthermore, the conventions of citation also vary. For instance, citations in e-Print use the first name of authors as only initial, while ones in DBLP use full names. All four data sets are pre-segmented (i.e., each field of co-authors, title, and venue are already known to us).

For the sampling technique, we used the implementation of [29] with a sample $S = 64$ and a threshold $\theta = 0.1$. Other remaining methods were implemented by us in Java. All experimentation was done using Microsoft SQL Server 2000 on Pentium III 3GHZ/512MB.

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

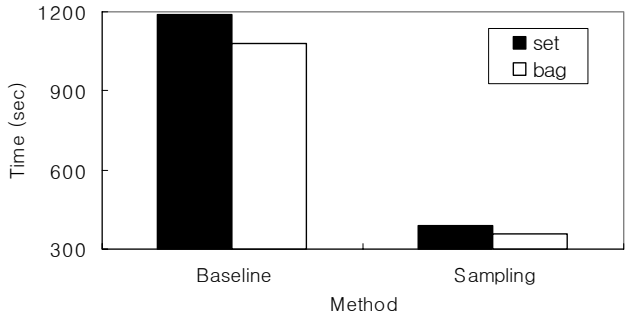
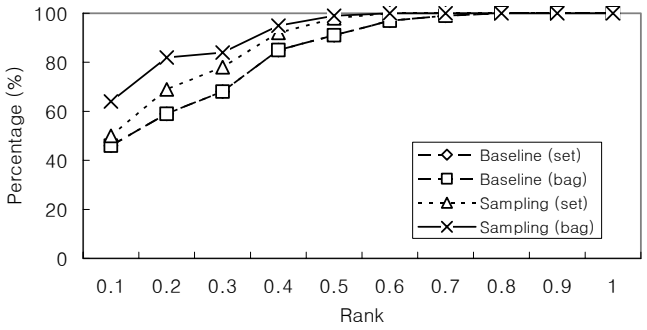


Figure 5.2: Scalability (EconPapers).

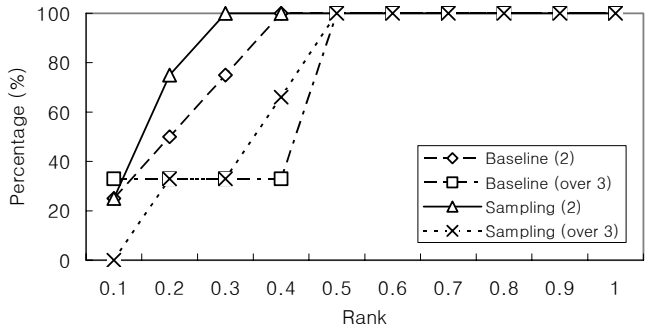
5.1.4 Experimental Results

Configuration. For this MER problem, we used two DLs as test-beds: DBLP and EconPapers. For DBLP (which authors know well), we collected real examples with the MER problem: e.g., Dongwon Lee, Chen Li, Wei Liu, Prasenjit Mitra, and Wei Wang, etc, and for EconPapers (which authors do not know well), we injected an artificial "false citations" into each author's citation collection. For both data sets, we tested how to find the "false citations" from an author's citations (that is, we had a solution set for both cases). In constructing token vectors, we used two models, *Set* and *Bag*, depending on the preservation of the multiple occurrences of the same token. For testing, we used the weights, $\alpha = 0.5$, $\beta = 0.3$, and $\gamma = 0.2$. As evaluation metrics, we used *time* for scalability, and *percentage/rank* ratio for accuracy (i.e., A false citation c_f must be ranked low in $sim(c_f, a)$. Thus, we measured how much percentage of false citations were ranked in the bottom 10%, 20%, etc).

Results. First, Figure 5.2 clearly shows the superior scalability of the sampling-based approach over the baseline one (about 3-4 times faster), regardless of set or bag models. Since the time complexity of the sampling-based approach is bounded by S, which was set to 64, for a large C such as DBLP, the scalability gap between two approaches further widens. Second, Figure 5.3(a) illustrates the accuracy of both approaches for EconPapers. For instance, when there is a single false citation c_f hidden in the 100 citations, the sampling approach with the bag model can identify c_f with over 60% accuracy (i.e., rank=0.1/%=64). Furthremore, when it



(a) EconPapers



(b) DBLP

Figure 5.3: Accuracy (EconPapers and DBLP).

can return upto 2 citations as answers, its accuracy improves to over 80% (i.e., rank=0.2/%=82). Since many tokens in citations tend to co-occur (e.g., same authors tend to use the same keywords in titles), the bag model that preserves this property performs better. Finally, Figure 5.3(b) shows results on DBLP using only the bag model. Note that some collection has a mixture of "2" authors' citations while others have that of "over 3" authors (e.g., there exists more than 3 authors with the same spellings of "Wei Liu"). Intuitively, collections with more number of authors' citations mixed are more difficult to handle. For instance, when 2 authors' citations are mixed, 100% of false citations are always ranked in the lower 30% (i.e., rank=0.3) using the sampling approach. However, when more than 3 authors' citations are mixed, the percentages drop to mere 35% – it is very difficult to decipher a false citation when it is hidden in a collection that contains a variety of citations from many authors.

5.1.5 Summary

The *Mixed Entity Resolution* problem is formally introduced and their solutions are explored. Since such a problem commonly occur in many of the existing bibliographic DLs, it is important to devise an effective and efficient solution to them. By utilizing one of the state-of-the-art sampling-based approximate join techniques, our solution is scalable yet highly effective. Furthermore, our proposal exploits associated information of author names (e.g., co-authors, titles, or venues) than names themselves.

5.2 Name Disambiguation using Multi-level Graph Partition (MGP)

In many applications, entities need to carry a unique identifier. This identifier can be as simple as a primary key in Databases or ISBN of books, or as complex as DNA fingerprint of people. When all applications adopt universal identifier system such as DOI, one does not have to worry about issues related with identifiers. However, in reality, it is not uncommon to find an application that uses non-unique data values as an identifier. One of commonly used such identifiers is a short *name description* ("names" in short hereafter) of entities. Examples include: name of persons, name of movies, name of cities, etc. Since these names of entities are not unique, inevitably, there can be multiple entities with the same names, causing a confusion. This problem is often referred to as the **Name Disambiguation** problem, where goal is to sort out the erroneous entities due to name homonyms.

Figure 5.4 illustrates the real case where entities are mixed due to their name homonyms. This shows screen-shots of home pages of two different "Dongwon Lee" returned from Google. In a sense, full names of users are used as a key for home pages.

In general, one can model the name disambiguation problem as *k-way clustering* problem. That is, given a set of mixed N entities with the same name description d, group N entities into k clusters such that entities within each cluster belong to the same real-world group (e.g., same author or movie). For instance, in Figure 5.4, one needs to group many web pages returned from Google for the query keyword

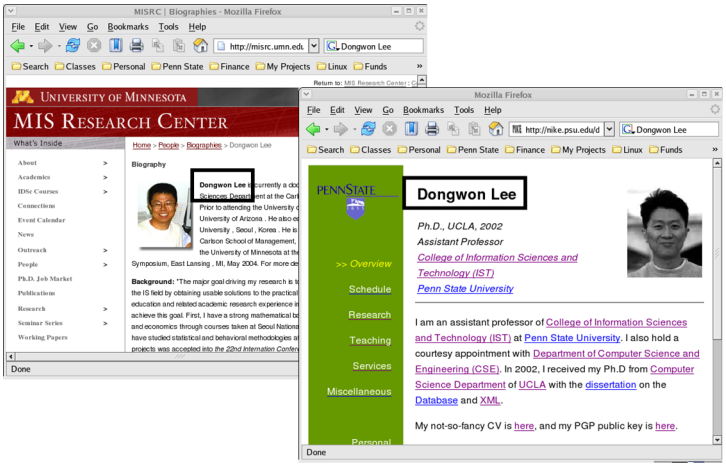


Figure 5.4: Examples of mixed entities due to homonyms – home pages of two different "Dongwon Lee" are returned from Google.

"Dongwon Lee" into two clusters – one for a faculty member at Penn State and the other for a graduate student at U. Minnesota.

In this problem, in particular, we study the scalability issue of the name disambiguation problem – when a large number of entities get un-distinguishable due to homonyms, how to resolve it? By and large, the scalability issue has been ignored in previous research of name disambiguation problem. Therefore, existing solutions tend to work well for a handful of mixed entities in the range of 10 or so, or a large number of entities with limited number of feature dimensions (e.g., [32, 4]). However, as data applications become more complicated and users increase rapidly, new needs arise to handle more large-scale name disambiguation problem. For instance, for a given "name" query keyword t (e.g., person, company, or movie), it is common to have thousands of web pages returned from search engines, all of which contain the keyword t and could have high dimension spaces in the vector space model. Therefore, it is important to have a scalable yet accurate name disambiguation algorithm.

For this goal, in this thesis, we first carefully examine two of the state-of-the-art solutions – k-way spectral clustering [32] and multi-way distributional clustering [4] – to the name disambiguation problem, and point out their limitations with respect to their scalability. Then, we adapt the *multi-level graph partition* technique to solve the large-scale name disambiguation problem. Our claim is empirically vali-

Name	Description
k	# of clusters
l	# of tokens
m	# of unique tokens (i.e., m-dimensional vectors)
n	# of documents (i.e., entities)
c	Constant

Table 5.1: Terms.

dated via experimentation – our proposal shows orders of magnitude improvement in terms of performance while maintaining equivalent or reasonable accuracy.

5.2.1 Name Disambiguation using MGP: Problem

Formally, using the terms of Table 5.1, the name disambiguation problem in our setting is defined as follows:

> Given a set of mixed entities E=$\{e_1, ..., e_n\}$ with the same name description d, group E into k disjoint clusters C=$\{c_1, ..., c_k\}$ ($k \leq n \leq m \leq l$) such that entities $\{e_p^i, ..., e_q^i\}$ ($1 \leq p \leq q \leq n$) within each cluster c_i belongs to the same real-world group.

Note that both n and k can be a substantially large number. In general, distance functions to measure the distance between two entities in the name disambiguation problem is more expensive than those used in conventional clustering framework. This is because each entity can be a long record or a whole document, instead of simple numeric or string values of attributes.

5.2.2 Name Disambiguation using MGP: Solution

The basic three-phased framework of our approach is illustrated in Figure 5.5:

In this section, we describe in details how to use MGP to solve the name disambiguation problem.

5.2.2.1 Graph Formation

We convert the given N entities into a graph $G = (V, E)$ as follows:

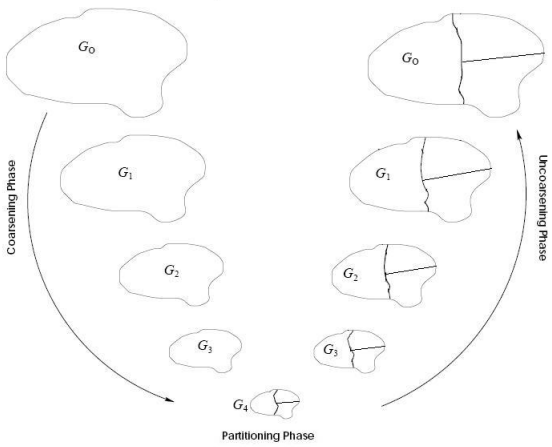


Figure 5.5: The three phases of the multi-level graph partition technique [42].

- Each entity e_i is mapped to a node $v_i(\in V)$.
- By treating the contents of an entity e_i as documents, we apply the standard vector space model to convert e_i into an m–dimensional vector (e.g., $X = (\alpha_1, ..., \alpha_m)$)[2]. If the i–th token in the entire token space appears in an entity e_i, then α_i is the TF/IDF[3] weight value of the i's token. Otherwise, $\alpha_i = 0$.
- Finally, edge weight between two entities x and y is computed as follows [14]: $TFIDF(x,y) = \sum_{w \in T_x \cap T_y} V(w,T_x) \times V(w,T_y)$, where (1) $V(w,T_x) = log(TF_{w,T_y} + 1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'}(log(TF_{w,T_y}+1) \times log(IDF_w))}}$ (symmetrical for $V(w,T_y)$), and (2) $V(w,T) = log(TF_{w,T}+1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'}(log(TF_{w,T}+1) \times log(IDF_w))}}$. In the TFIDF similarity function, TF_{w,T_x} is the frequency of w in T_x, and IDF_w is the inverse of the fraction of names in a corpus containing w;

[2]Let us assume that a standard stemming process has been done to filter out stop words.

[3]By definition, a weight is a certain value normalized in terms of importance of a word token in a document. The *Term Frequency* (TF) is a measure of the importance of the term t in a particular document. Therefore, if a term t appears in a particular document frequently, the TF weight of the term t will be high. On the other hand, *Inverse Document frequency* (IDF) is a measure of importance across documents in a collection. If a term t appears infrequently in a collection, the IDF weight will be high.

5.2.2.2 Multi-level Graph Partition

The graph partitioning problem [41] is to cluster the given graph into *equally sized* sub graphs among which the number of edges is optimized. However, as the size of a graph is significantly huge, most graph partitioning algorithms can take a large amount of time. Thus, to speed up graph partitioning algorithms but yet minimize the number of edge-cuts, a multi-level graph partitioning algorithm, METIS [42], was proposed in the parallel computing community as follows: (1) The given graph is condensed into the smallest graph in which thare are only a few vertices; (2) The vertices in the smallest graph are clustered by a clustering algorithm; and (3) The clustered graph is magnified into the size of the original graph. When the size of sub graphs is nearly equal, METIS is an efficient partitioning method on very large graphs. However, equally sized partitions by METIS are not desired in many other domains. To surmount such resriction of METIS, Dhillon et al. [20] proposed a weighted kernel k–means algorithm. Formally, the multi-level graph partitioning algorithm works as follows:

- **The coarsening phase.** The original graph G is successively subdivided into smaller graphs $G_1, G_2, ..., G_k$ such that $|V| > |V_1| > ... > |V_k|$, where level $i \in [1..k]$ and $|V_i|$ is the number of vertices in graph G_i. In G_i, visit each vertex randomly, and then merge a vertex v with a neighbor w that maximizes the edge weight between v and w. Once all the vertices in G_i are visited, the coarsening process at level i is completed.

- **The partitioning phase.** Through the repeated coarsening steps, the smallest graph is determined such that the size of the graph is less than $20 \times k$. Then, the spectral algorithm of Yu and Shi [72] is performed to cluster the smallest graph.

- **The uncoarsening phase.** In order to derive partitions of the original graph G, the uncoarsening step is required in which the clustered graphs are repeatedly projected to larger graph. Suppose that the size of G_i was decreased into that of G_{i+1} in the coarsening phase. Furthermore, two vertices v and w of G_i were binded to a single vertex (v, w) of G_{i+1}. Then, the vertex (v, w) was partitioned into a cluster c in the partitioning phase. In

the uncoarsening phase, the vertices v and w can be grouped to the same cluster c. Then, more accurate projection to the larger graph is performed using a weighted kernel k-means uncoarsening algorithm. According to [20], graph clustering objective functions (e.g., Ratio Association) can be transformed into weighted kernel k–means objectives, with the weight of vertex w and kernel matrix K, to locally optimize these graph clustering objectives. Therefore, given a similarity matrix M, the kernel matrix K of a graph clustering objective function (i.e., Ratio Association) is computed by $\sigma I + M$ where σ is a real number. Subsequently, compute the updated clusters as $argmin_k(K_{xx} - \frac{2\times\Sigma_{y\in\pi_k^i}\times w_y\times K_{xy}}{\Sigma_{y\in\pi_k^i}\times w_y} + \frac{\Sigma_{y,z\in\pi_k^i}\times w_y\times w_z\times K_{yz}}{(\Sigma_{y\in\pi_k^i}\times w_y)^2})$, where π_k^i is the k-th cluster in the i-th iteration.

5.2.3 Two State-of-the-art Solutions: MDC & SC

5.2.3.1 Multi-way Distributional Clustering (MDC)

Bekkerman and McCallum used the *multi-way distributional clustering (MDC)* to solve the name disambiguation problem in [4]. We briefly describe about MDC here.

Given a set of k clusters, $c_1, ..., c_k$, all tokens of $c_{i\in[1..k]}$ are placed in a *single* cluster while each entity $c_{i\in[1..k]}$ is placed in a *singleton* cluster. For instance, given a set of word tokens and documents in a collection, all the tokens are put in a single cluster while each document in the collection is assigned to each singleton cluster. Then, during top-down/bottom-up clustering iterations, the top-down clustering scheduler splits each element (e.g., a word) uniformly at random to two sub-clusters. On the other hand, the bottom-up clustering scheduler merges each element (e.g., a document in a collection) with its closest neighboring cluster. The scheduling scheme is pre-determined in MDC.

For example, if a schedule scheme is "words, words, words, documents, documents," three clustering iterations over words will be processed first, followed by two iterations over documents. Finally, for all elements, correct clusters are created based on Mutual Information – that is, it correctly clusters a random variable X (e.g., documents) by a joint probability distribution between X and an observed variable Y (e.g., words). The joint probability distribution is computed based on

a table summarizing # of occurrences of times $x \in X$ occurred with $y \in Y$ (e.g., # of times a term y appears in a document x).

5.2.3.2 *k*-way Spectral Clustering (SC)

Han et al. used the *k-way spectral clustering (SC)* to solve the name disambiguation problem in [32]. Here, we again briefly explain the SC.

Consider a set of entities E. The spectral clustering methods consider the similarity matrix S, where $S_{i,j}$ is a similarity measure between entities $e_p, e_q \in E$. As one of commonly used spectral clustering methods, *Shi-Malik* algorithm [62] partitions entities into two sets based on eigenvector v corresponding to the second smallest eigenvalue (i.e., Fiedler vector) of the Laplacian of S. Similarly, the *Meila-Shi* [53] and Han et al. [32] use the eigenvectors corresponding to k largest eigenvalues of the matrix $P = DS^{-1}$ for k, and then cluster entities using k-means or pivoted QR decomposition by their respective k components in eigenvectors.

5.2.3.3 Computational Complexity

The MDC method iteratively performs agglomerative clustering over terms (e.g., word tokens) and conglomerate clustering over documents (e.g., web pages or citations in a collection) at random, and assigns documents to more accurate clusters based on the joint probability distribution of terms and documents. Thus, this algorithm is significantly expensive on large-scale data.

For instance, suppose that the MDC method has two clustering systems X and Y. X is the agglomerative clustering system over tokens such that a cluster $x \in X$ and an element $e_i \in X_{i=1..l}$. Y is the conglomerative clustering system over documents such that a cluster $y \in Y$ and an element $e_i \in Y_{i=1..n}$. Since the MDC method focuses on clustering documents, the maximal number of iterations to obtain the final clusters is $O(logn)$. During each iteration, each cluster in X is randomly split to two equally sized sub clusters and then each token $e_i \in x_i$ is correctly placed into x_j based on Mutual Information. Next, each cluster in Y is randomly merged to its nearest neighbor cluster and *cluster corrections* are performed to minimize the Bayes classification error. At each iteration in the top-down step, entity e_i is placed into a cluster x_j such that Mutual Information

$I(X, Y)$ is maximal. Similarly, the same process is performed in the bottom-up step. Therefore, the computational complexity of MDC is:

$$O(l \cdot n \cdot logn)$$

On the other hand, in the k-way SC algorithm, given a similarity matrix M, $v_1, ...v_k$ eigenvectors of M are computed by the k largest eigenvalues to create the matrix V with k columns of eigenvectors of M. Finally, the rows of V are clustered. In general, the running time of these spectral clustering algorithms is dominated by the computation of eigenvectors of M, and the complexity time is known as [58, 33, 59, 26]:

$$O(\frac{4}{3} \cdot c \cdot m^3) \approx O(m^3)$$

As clearly shown here, since both MDC and SC have quadratic and cubic time complexities, they do not scale well. The *multi-level graph partition (MGP)* [42, 20] (to be elaborated in Section 5.2.2) consists of three steps. During the coarsening step, the size of the graph is repeatedly decreased; in the clustering step, the smallest graph is partitioned; and during the uncoarsening step, partitioning is successively refined to the larger graph. During the coarsening step, since the size of the graph is decreased from level to level and all the vertices in the graph are visited at each level, the complexity gets $O(\log n)$. In the clustering step, if we use one of spectral algorithms, the complexity is $O(\frac{4}{3} \cdot c \cdot \{20 \cdot k\}^3)$. During the uncoarsening step, the running time at each level is $O(nz)$, where nz is # of non-zero entries in the kernel matrix. Overall, therefore, the computational complexity of the MGP is:

$$O(\log n + \frac{4}{3} \cdot c \cdot (20 \cdot k)^3 + \log n) \approx O(k^3)$$

In conclusion, since $k \ll n$, the computational complexity of the MGP is the most efficient, compared to that of MDC and of SC.

5.2.4 Experimental Set-up

For the MDC, k-way SC, and MGP methods, we used the implementation of [3], [68], and [19], respectively. For the implementation of TF/IDF Cosine similarity, we used SecondString [65]. All experimentation were done on 4×2.6Ghz Opteron processors with 32GB of RAM.

Data sets.

For validation, we have used four data sets – two small and two large data sets from real examples. Figure 5.6 illustrates the overall statistics of four data sets.

- First, the `ACM-s` is real test case that we have gathered from the ACM digital library. When two scholars have the same name spellings, their citation data are mistakenly merged into a single collection, leading to an incorrect citation analysis results. For instance, Figure 5.4(a) illustrates a collection of mixed citation by four "Wei Wang" in DBLP. We collected 24 real examples as shown in Table 5.2, and manually checked their correctness.

- The `WWW-s` is a small-sized test case using the 1,085 web pages that [4] used. In 2004, [4] extracted 12 personal names from Melinda Gervasio's email directory. Then, 100 top-ranked web pages of each name were retrieved from Google, and cleaned and manually labeled by authors. The resulting data set consists of 1,085 web pages, 187 different persons, and 420 relevant pages. Table 5.3 shows the statistics of the data set. For instance, when "Tom Mitchell" is issued as a query to Google, 92 web pages are retrieved. Among these 92, there are 37 namesakes to "Tom Mitchell". For example, among 92 web pages, "Tom Mitchell" appears as musicians, executive managers, an astrologist, hacker, and rabbi – 32 different kinds. That is, a set of 32 entities are mixed since they all have the same name description of "Tom Mitchell". Like `ACM-s`, `WWW-s` is a small but high-quality test case with more number of clusters per case.

- Next, the `DBLP-m` is a medium-sized citation test case generated from DBLP digital library. To generate an ambiguous name data set, we clustered author names from the entire DBLP citation data [4] if two authors share the same

[4]DBLP has about 360,000 authors and 560,000 citations in it.

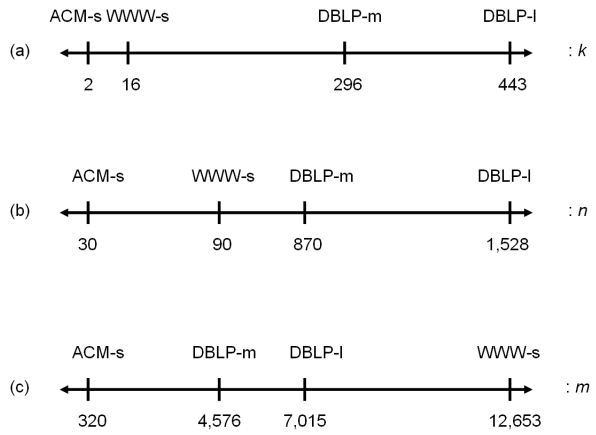


Figure 5.6: `Overview of statistics of test cases: (a) average` k `(b) average` n `(c) average` m.

first name initial and full last name. Then, we sorted the formed name clusters by the number of name variants. Finally, we obtained top-10 ambiguous names. For instance, # of "J. Lee" variants is 421 (top ranked), # of "S. Lee" variants is 391 (2nd ranked), # of "J. Kim" variants is 377 (3rd ranked) and so. For the details, please refer to Table 5.4.

- Finally, `DBLP-l` is a large-scale citation test case, similar to `DBLP-m`, except that this time only the full last name is used in the blocking. Table 5.5 shows the statistics of the data set.

Evaluation Metrics.

To evaluate competitive clustering methods, each cluster $c_i \in C_{i=1,..,k}$ is assigned with the most dominant label in c_i. Then, we measure the *precision* and *recall* for c_i as follows [64]:

$$Precision(c_i) = \frac{\Sigma_{i=1}^{i=k}\alpha(c_i)}{\Sigma_{i=1}^{i=k}(\alpha(c_i) + \beta(c_i))}$$

$$Recall(c_i) = \frac{\Sigma_{i=1}^{i=k}\alpha(c_i)}{\Sigma_{i=1}^{i=k}(\alpha(c_i) + \gamma(c_i))}$$

where $\alpha(c_i)$ denotes # of entities correctly assigned to c_i, $\beta(c_i)$ denotes # of entities incorrectly assigned to c_i, and $\gamma(c_i)$ denotes # of entities incorrectly not assigned to c_i.

Name data set	k	n	m
H Cai	2	5	89
Wei Cai	2	7	120
John M. Carroll	2	92	673
Li Chen	2	60	718
Yu Chen	2	46	594
Hui Han	2	15	184
Youngjae Kim	2	3	62
Dongwon Lee	2	30	322
Chen Li	2	31	343
Jia Li	2	27	276
Jian Li	2	21	284
Lin Li	2	11	145
Peng Liu	2	32	344
Wei Liu	2	43	530
Zhenyu Liu	2	8	139
Jiebo Lou	2	34	311
Murali Mani	2	11	131
Prasenjit Mitra	2	11	115
Sanghyun Park	2	18	201
Hui Song	2	6	79
James Ze Wang	2	33	310
Wei Wang	4	143	1,264
Yuan Xie	2	20	210
Wei Xu	2	17	230
Average	2	30	320

Table 5.2: Statistics of test case `ACM-s`.

Table 5.6 illustrates an example of clustered documents. In "Cluster 2", since the most dominent label is *SriEng*, "SriEng" is assigned as the class label of the second cluster. $\alpha(c_2) = 3$, $\beta(c_2) = 2$, and $\gamma(c_2) = 1$. Therefore, the precision of Cluster 2 is $\frac{\alpha(c_2)}{\alpha(c_2)+\beta(c_2)} = \frac{3}{3+2} = 0.6$, and the recall of Cluster 2 is $\frac{\alpha(c_2)}{\alpha(c_2)+\gamma(c_2)} = \frac{3}{3+1} = 0.75$.

5.2.5 Experimental Results

First, let us consider how accurate three methods are. Table 5.7 shows precisions of MDC, SC, and MGP in four test cases. MGP shows better average precisions than both SC and MDC for three cases. On the other hand, SC is not a straightforward

Name data set	k	n	m
Adam Cheyer	2	97	12,146
William Cohen	10	88	9,036
Steve Hardt	6	81	14,088
David Israel	19	92	11,739
Leslie Pack Kaelbling	2	89	12,153
Bill Mark	8	94	10,720
Andrew McCallum	16	94	11,166
Tom Mitchell	37	92	10,356
David Mulford	13	94	16,286
Andrew Ng	29	87	10,441
Fernando Pereira	19	88	10,999
Lynn Voss	26	89	22,706
Average	16	90	12,653

Table 5.3: Statistics of test case `WWW-s`. Note here n (i.e., # of entities) is in fact # of web pages.

Name data set	k	n	m
C. Chen	220	787	4,129
Y. Chen	238	853	4,689
H. Kim	290	713	3,931
J. Kim	377	1,104	5,567
S. Kim	302	847	4,469
Y. Kim	240	559	3,376
C. Lee	234	676	3,842
H. Lee	242	557	3,509
J. Lee	421	1,281	6,234
S. Lee	391	1,320	6,011
Average	296	870	4,576

Table 5.4: Statistics of test case `DBLP-m`.

method as shown in Table 5.7. Overall, the larger the data size gets, the poorer the precision becomes. This is because the name data sets of these three methods are clustered into multi classes. Note that the precision of MGP in the `WWW-s` test case and that of MDC in the `ACM-s` test case. According to Table 5.7, MDC is a better name disambiguation method than MGP for web page data set, but it becomes the opposite case for citation data set. This indicates that using TF/IDF cosine similarity to obtain edge weights between vertices (e.g., web pages or citations) is

Name data set	k	n	m
Brown	416	1,233	6,611
Chan	478	1,310	6,225
Cheng	451	1,508	6,936
Johnson	437	1,630	7,604
Jones	398	1,561	6,869
Lu	471	1,581	7,071
Martin	398	1,400	7,489
Wong	450	1,730	7,022
Xu	485	1,799	7,494
Zhou	441	1,532	6,824
Average	443	1,528	7,015

Table 5.5: Statistics of test case `DBLP-1`.

```
Cluster 1:
           ID#1_Other.txt
Cluster 2:
           ID#63_SriEng.txt
           ID#85_SriEng.txt
           ID#39_Player.txt
           ID#1_Lawyer.txt
           ID#40_SriEng.txt
Cluster 3:
           ID#6_PSUProf.txt
           ID#8_SriEng.txt
```

Table 5.6: An example of clustered documents (i.e., entities) with the format of ID#*document ID_document label.txt* [64].

more effective in the citation data set. Intuitively, there is likely to be stronger relationship among an author and its variants than web page data sets. That is, a document contains a number of terms only a few of which can be considered to be important to identify variants.

Table 5.8 illustrates the average recall of three methods. For the small data sets, SC is the winner while for the large data sets, MDC is the winner – MGP is always the 2nd. While MGP performs coarsening and un-coarsening steps for the initial graph partitioning, the graph is approximately transformed to smaller sized one. These processes can decrease the recall of MGP in the large-scale test cases.

Test Case	MDC	SC	MGP
ACM-s	0.84	0.82	**0.86**
WWW-s	**0.73**	0.59	0.65
DBLP-m	0.43	0.06	**0.54**
DBLP-l	0.4	0.05	**0.44**

Table 5.7: Average precision.

Test Case	MDC	SC	MGP
ACM-s	0.57	**0.82**	0.64
WWW-s	0.36	**0.57**	0.36
DBLP-m	**0.35**	0.06	0.34
DBLP-l	**0.3**	0.05	0.24

Table 5.8: Average recall.

Please note that MGP outperforms MDC in precision but MDC shows better recall than MGP. This indicates that there exists a trade-off between MGP and MDC in clustering. In conclusion, although MGP is not the clear winner for all test cases in both precision and recall, it appears to show pros of both approaches. This can be clear in Figure 5.7 showing F-measure (i.e., a harmonic sum of precision and recall) of both small test cases and Figure 5.7 of both large test cases.

Figure 5.8 (a)-(c) illustrates F-measure changes with the variation of n in three test cases, ACM-s, DBLP-m, and DBLP-l, respectively. Similarly, Figure 5.8 (d)-(f) shows F-measure changes with the variation of m in the test cases. Note that as # of citations and # of dimensions of a vector are increased, F-measures are decreased considerably. For instance, Figure 5.8 (b), in case of $n = 500$, MGP shows about 0.6 as the F-measure. On the other hand, when $n = 900$, F-measure is less than 0.4. Figure 5.8 (c) shows the similar pattern to (b). In addition, as shwon in Figure 5.8 (d) and (e), when # of dimensions of a vector is increased, the F-measures are reduced. In particular, according to Figure 5.8, the SC method is the worst, compared to MDC and MGP.

Table 5.9 shows running time of three methods for four test cases. It is clear that MGP is always winner, as we have predicted in Section 3.3. Note that both DBLP-m and DBLP-l are large-scale test cases. In DBLP-m, MGP is 157 times faster than SC, and in DBLP-l 383 times faster than MDC. Even if WWW-s is a small sized

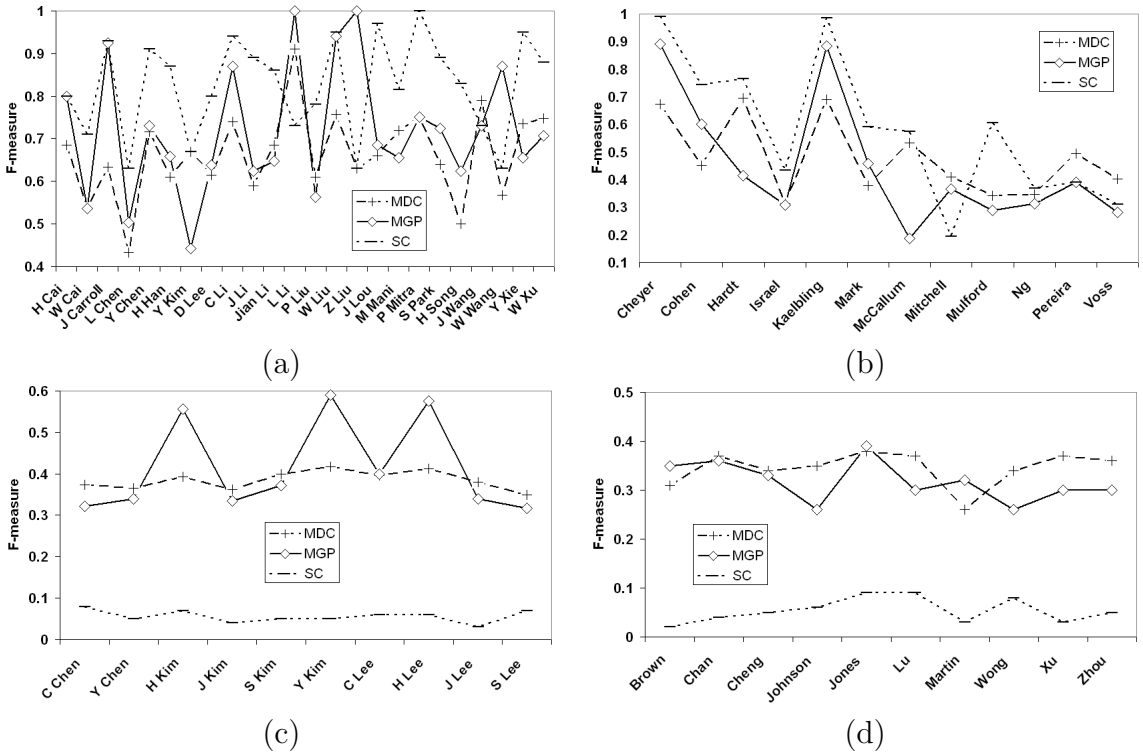


Figure 5.7: F-measure of (a) `ACM-s`, (b) `WWW-s`, (c) `DBLP-m`, and (d) `DBLP-l`.

Test Case	MDC	SC	MGP
`ACM-s`	0.12	2.2	**0.0**
`WWW-s`	2.3	4,274	**0.0**
`DBLP-m`	74	77	**0.49**
`DBLP-l`	609	169	**1.59**

Table 5.9: Running time (in sec.).

data set, the running time of SC is significantly large – 4,274 sec. This is because # of dimension per vector in `WWW-s` is considerably large. For instance, the average # of dimensions a vector in `WWW-s` is about 13K while in `DBLP-l`, the largest data set, there is on average 7K dimensions per vector. Note that, unlike MDC and MGP, k-way SC method compute k eigenvectors of a matrix. Thus, as the number of dimensions of the matrix increases, SC takes considerably more time to identify and categorize name variants. Unexpectedly, MDC is the slowest method, even worse than SC in `DBLP-l`. This is because MDC considers # of words as its input data while SC and MGP use # of documents. Thus, the input size of MDC is

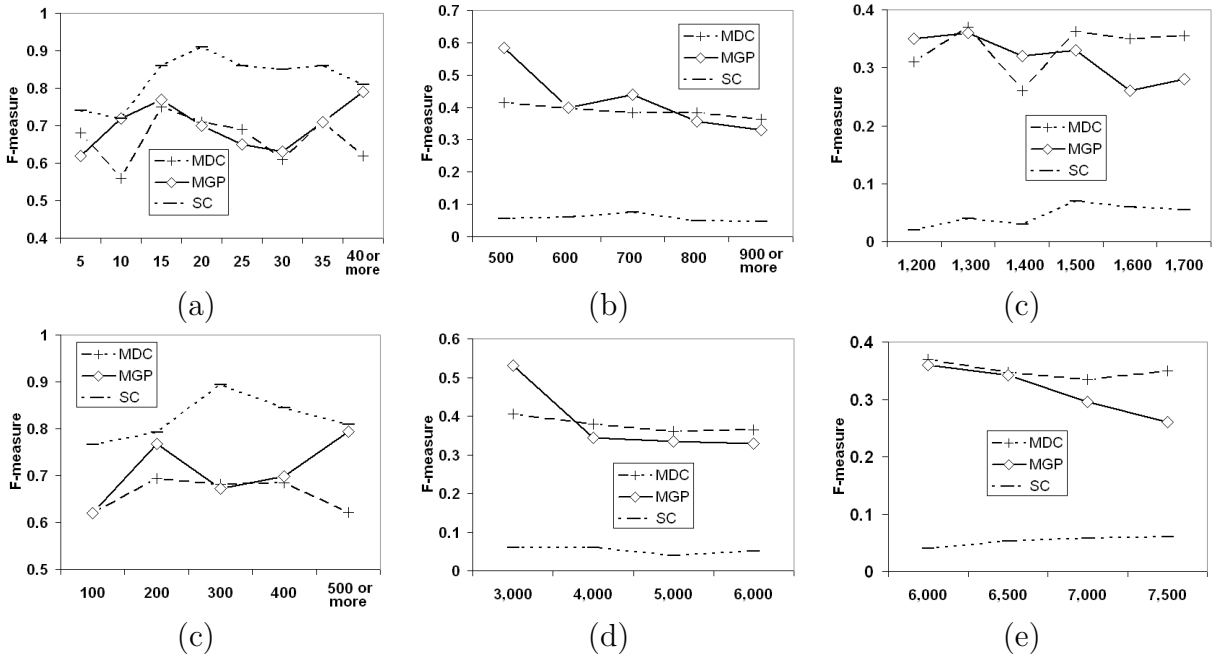


Figure 5.8: (a-c) F-measure changes with the variation of n in `ACM-s`, `DBLP-m`, and `DBLP-l`; (d-f) F-measure changes with the variation of m in `ACM-s`, `DBLP-m`, and `DBLP-l`.

significantly larger than that of SC and MGP.

In summary, when MGP is used in the name disambiguation problem, it shows better precision and equivalent or slightly worse recall than both MDC and SC. However, MGP outperforms the other two in terms of scalability by orders of magnitude (upto 383 times).

5.2.6 Summary

In this chapter, we have studied the name disambiguation problem and their solution. In particular, we pointed out the limitations of two state-of-the-art methods to the name disambiguation problem, and instead proposed to use multi-level graph partition techniques. Our proposal is empirically validated at the end.

As to future direction, we plan to improve recall and precision of MGP-based Name Disambiguation technique.

Chapter 6

Conclusions and Future Research

6.1 Contributions

In this work, we have investigated three specialized types of the Entity Resolution (ER) problems: (1) the Split Entity Resolution (SER) problem; (2) the Grouped-Entity Resolution (GER) problem; and (3) the Mixed Entity Resolution (MER) problem. For each type of problems, the goal of our research is to develop a set of generic domain-independent, effective, efficient data cleaning solutions in terms of compatibility with SQL and Database engine.

Toward this goal, our dissertation work is centered around novel, scalable solutions using *Blocking-based Framework*, *Sampling-based Approximated Join*, and *Multi-level Graph Partition*. Especially, for the GER problem, we have developed two graph theoretic algorithms, *Quasi-Cliques* and *Group Linkage* based on Bipartite Matching.

In detail, some of the major contributions of this research are as follows:

1. **The Split Entity Resolution (SER) problem.** we developed a *scalable two-step framework*, in which step 1 is to substantially reduce the number of candidates via blocking, and step 2 is to measure the distance of two entities via collective knowledge. Through this framework, we comparatively studied various approaches (e.g., supervised methods, distance functions, and vector-based methods) to identify and correct name variants.

2. **The Grouped-Entity Resolution (GER) problem.** Unlike the previ-

ous approaches, relying on textual similarity, producing a large number of false positives, we presented the experience of applying a recently proposed graph mining technique, *Quasi-Clique*, atop conventional entity resolution solutions. This approach exploits contextual information mined from the group of elements per entity in addition to syntactic similarity. In addition, we focus on the intuition that two groups can be linked to each other if there is high enough similarity between matching pairs of individual records that constitute the two groups and there is a large fraction of such matching record pairs. Formalizing this intuition, we proposed a *Group Linkage measure* based on bipartite graph matching with better accuracy than the existing textual similarity measures.

3. **The Mixed Entity Resolution (MER) problem.** Although the existing clustering and classification methods are not scalable, we propose a *scalable citation labeling algorithm* using the sampling-based approximated join algorithm to quickly determine a small number of candidates from the entire entities. In addition, we proposed a *scalable name disambiguation using multi-level graph partition*. In general, although k-way spectral clustering algorithm is the most effective, it tends to take a large amount of time as the size of a graph is considerably huge. To speed up such a graph partitioning algorithm but yet optimize clusters, we applied multi-level graph partitioning algorithm to this MER probelm.

6.2 Limitations and Assumptions

A more critical analysis of this work can be carried out by discussing limitations and assumptions of this work.

1. For the data cleaning problem, the research direction is as follows: The crawling softwares take documents from several digital libraries. Then, using some algorithms, meta data are extracted and raw citation data are given to make clean redundant data. We have not researched how to extract meta data and raw citation data. Thus, after we assume that raw citation data are given, we start to study a variety of solutions to SER, GER, and MER

problems in our work. Subsequently, we need to develop system support to exploit the clean knowledge. The final purpose of this research is to build a prototype system.

2. **Two-step SER Framework.** In our context, we only focus on exploiting co-author information as the associated information of an author. However, Han et al. [31] reported a promising result by using co-authors, titles, venues, or even hybrid of them as additional information. Therefore, one possible area of extension is to support other associated information to our two-step framework.

3. **Quasi-Clique based Distance Measure.** We plan to examine how to overcome the shortcoming to make the superimposition technique generic: As we have demonstrated, our proposal shows consistent improvement in both precision and recall on various data sets. However, there is the limitation that we need to overcome. In mining context graphs, we require the *base graph*. For data sets with rich semantics such as citation data set, we can use domain-specific graphs as the base graph (e.g., collaboration graph from all co-authors). Otherwise, when such domain-specific graphs are not available, we should use the token co-occurrence graph as the base graph, but the contribution of our distQC decreases. In essence, this make sense – the more semantics are provided in the base graph, the better " context graphs" can be made.

4. **Sampling-based Citation Labeling Method.** According to our experimental results, when more than 3 authors' citations are mixed, our citation labeling algorithm shows poor performance. Therefore, we need to build a solution to remedy this complicated problem. The MER problem can be naturally cast to K-Clustering Problem (i.e., Cluster N data points into K clusters). Such clustering methods are proposed by Han et al. [31] and Malin [51]. Therefore, we need to compare their methods to ours.

6.3 Future Research

First, we can extend current work to address issues discussed in Chapter 6.2. Furthermore, some of the possible future research extensions are as follows:

1. **Clean Databases**

   To achieve clean databases, improving the quality of data must be an important research issue in modern days. This *Data Cleaning* problem commonly occurs in a wide veriety of applications – information integration, data warehouse, digital libraries, search engines, e-commerce, census surveys, image processing, bio medical informatics, and so forth. In addition, there are many challenges to handle: (1) Improving the existing cleaning algorithms; (2) Constructing effective evaluation tools; (3) High performance record linkage; and (4) Improving missing value estimation.

2. **Domain Ontologies for Web Services**

   Recently, the *semantic* web services technology is the main tool to automatically build domain ontologies in the context of the World Wide Web. However, since individual web services tend to be created in isolation, there exist variant affairs over different sources in terms of different formats, abbreviation, typographical, or homonym (e.g., "cost" and "price"). That is, the same concepts and instances may be represented quite differently over heterogeneous web service sources. Therefore, the major challenge is to identify the semantic correspondences between concepts and instances retrieved from different sources.

3. **Semantic Web Services Discovery and Composition Problem**.

   When no single web service satisfies the given request fully, one needs to compose multiple web services to fulfill the goal. In the web services discovery and composition problem, to determine when an operation in a web service can invoke another application in other web service, it should be considered if their corresponding input and output parameters are "lexicographically" matching. But since individual web services are often created in isolation,

matching the parameters is non-trivial due to different formats, abbreviation, typographical, or homonym (e.g., "cost" and "price"). To handle such web service environment, we can consider approximate matching approaches. Currently, majority of public web services need to have annotated semantics.

4. **Information Integration and Security**

   In the database community, the issue for sharing minimal information across private databases will be promising. For instance, consider two credit card companies that wish to identify fraudulent customer list common to both companies. However, two companies do not want to share their customer-related information with the other party. In this problem, one faces two challenges of how to design the proper security model of information integration and how to detect/consolidate redundant data occurred by likely differences in conventions among private databases.

5. **Advanced Search Engines**

   (a) *Challenge 1.* A search engine (e.g., PaperFinder) regularly queries new articles to several digital libraries such as ACM and DBLP. However, such a system usually return different formats of citations because of diverse citation formats in digital libraries (e.g., "Russell S, Norvig P (1995) Artificial Intelligence: A Modern Approach, Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey" and "[RN95] Artificial Intellignece-aModern Approach by S. Russell and P. Norvig. Prentice Hall International, Englewood Cliffs, NJ, USA, 1995"). This problem commonly occurs in other popular scientific search engines such as CiteSeer and Google Scholar as well. Therefore, one needs to develop system support to exploit the clean knowledge in search engines.

   (b) *Challenge 2.* It is known that about 30% of queries include person names and 100 million persons share only about 90,000 person names. This indicates that the search results are a mixture of web pages about different people with the same name spellings. Thus, ideal search engines should cluster web pages by different people sharing the same name.

Bibliography

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

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

[3] R. Bekkerman. "Multi-way Distributional Clustering (MDC)", 2005. http://www.cs.umass.edu/ ronb/mdc.html.

[4] R. Bekkerman and A. McCallum. "Disambiguating Web Appearances of People in a Social Network". 2005.

[5] O. Benjelloun, H. Garcia-Molina, Q. Su, and J. Widom. "Swoosh: A Generic Approach to Entity Resolution". Technical report, Stanford University, 2005.

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

[7] Digital Bibliography and Library Project (DBLP). http://dblp.uni-trier.de/.

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

[9] Bond. http://bond.unleashedinformatics.com.

[10] V. R. Borkar, K. Deshmukh, and S. Sarawagi. "Automatic Segmentation of Text into Structured Records". Santa Barbara, CA, May 2001.

[11] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. "Robust and Efficient Fuzzy Match for Online Data Cleaning". 2003.

[12] S. Chaudhuri, V. Ganti, and R. Kaushik. "A Primitive Operator for Similarity Joins in Data Cleaning". In *IEEE ICDE*, 2006.

[13] W. Cohen. "Data Integration Using Similarity Joins and a Word-based Information Representation Language". 18(3):288–321, 2000.

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

[15] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *"Combinatorial Optimization (2nd Ed.)"*. Prentice-Hall, 1997.

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

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

[18] P. T. Davis, D. K. Elson, and J. L. Klavans. "Methods for Precise Named Entity Matching in Digital Collection". 2003.

[19] I. Dhillon, Y. Guan, and B. Kulis. "graclus software", 2005. http://www.cs.utexas.edu/users/dml/Software/graclus.html.

[20] I. S. Dhillon, Y. Guan, and B. Kulis. "A Fast Kernel-based Multilevel Algorithm for Graph Clustering". 2005.

[21] X. Dong, A. Y. Halevy, and J. Madhavan. "Reference Reconciliation in Complex Information Spaces". 2005.

[22] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. "Duplicate Record Detection: A Survey". 19(1):1–16, Jan. 2007.

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

[24] ACM: Association for Computing Machinery. http://www.acm.org/.

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

[26] G. H. Golub and G. F. Van Loan. "Matrix Computations", June 1996. http://www.press.jhu.edu/.

[27] Goole. http://www.google.com.

[28] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. "Approximate String Joins in a Database (Almost) for Free". In *VLDB*, pages 491–500, 2001.

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

[30] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. "Merging the Results of Approximate Match Operations". In *VLDB*, pages 636–647, 2004.

[31] H. Han, C. L. Giles, and H. Zha et al. "Two Supervised Learning Approaches for Name Disambiguation in Author Citations". June 2004.

[32] H. Han, H. Zha, and C. Lee Giles. "Name Disambiguation in Author Citations using a K-way Spectral Clustering Method". June 2005.

[33] B. Hendrickson and R. Leland. "An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations". Technical report, Tech. rep. SAND92-1460, Sandia National Lab., Albuquerque, 1992.

[34] M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". 1995.

[35] Y. Hong, B.-W. On, and D. Lee. "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach". Bath, UK, September 2004.

[36] D. Hull. "Using Statistical Testing in the Evaluation of Retrieval Experiments". pages 329–338, Pittsburgh, 1993.

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

[38] IMDB. http://www.imdb.com.

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

[40] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. "Exploiting Relationships for Domain-independent Data Cleaning". In *SIAM Data Mining (SDM) Conf.*, 2005.

[41] G. Karypis and V. Kumar. "Multilevel k-way Partitioning Scheme for Irregular Graphs". 48(1):96–129, 1998.

[42] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular graphs". 20(1):359–392, 1999.

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

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

[45] D. Lee, B.-W. On, J. Kang, and S. Park. "Effective and Scalable Solutions for Mixed and Split Citation Problems in Digital Libraries". In *ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS)*, June 2005.

[46] M. L. Lee, W. Hsu, and V. Kothari. "Cleaning the Spurious Links in Data". 19(2):28–33, 2004.

[47] M. Ley. "The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives". Lisbon, Portugal, September 2002.

[48] Jia Li, James Z. Wang, and Gio Wiederhold. "IRM: Integrated Region Matching for Image Retrieval". pages 147–156, Los Angeles, CA, Oct. 2000.

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

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

[51] B. Malin. "Unsupervised Name Disambiguation via Social Network Similarity". In *SIAM SDM Workshop on Link Analysis, Counterterrorism and Security*, 2005.

[52] A. McCallum, K. Nigam, and L. H. Ungar. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". Boston, MA, August 2000.

[53] M. Meila and J. Shi. "A Random Walks View of Spectral Segmentation". 2001.

[54] N. Paskin. "DOI: a 2003 Progress Report". 9(6), June 2003.

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

[56] J. Pei, D. Jiang, and A. Zhang. "On Mining Cross-Graph Quasi-Cliques". August 2005.

[57] J. Pei, D. Jiang, and A. Zhang. "On Mining Cross-Graph Quasi-Cliques". August 2005.

[58] A. Pothen, H. D. Simon, and K.-P. Liou. "Partitioning Sparse Matrices with Eigenvectors of Graphs". 11:430–452, 1990.

[59] A. Pothen, H. D. Simon, L. Wang, and S. T. Bernard. "Toward a Fast Implementation of Spectral Nested Dissection". pages 42–51, 1992.

[60] S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". 2002.

[61] W. Shen, X. Li, and A. Doan. "Constraint-Based Entity Matching". In *AAAI*, 2005.

[62] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation". 22(8):888–905, 2000.

[63] SiMETRICS. http://www.dcs.shef.ac.uk/ sam/stringmetrics.html.

[64] N. Slonim, N. Friedman, and N. Tishby. "Unsupervised Document Classification using Sequential Information Maximization". 2002.

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

[66] M. M. M. Synman and M. J. van Rensburg. "Revolutionizing Name Authority Control". 2000.

[67] Uniprot. http://www.pir.uniprot.org/cgi-bin/upEntry?id=P38910.

[68] D. Verma and M. Meila. "spectral clustering toolbox", 2003. http://www.ms.washington.edu/ spectral/.

[69] James Z. Wang, Jia Li, and Gio Wiederhold. "SIMPLIcity: Semantics-sensitive Integrated Matching for Picture Libraries". 23(9):947–963, 2001.

[70] J. W. Warnner and E. W. Brown. "Automated Name Authority Control". 2001.

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

[72] S. X. Yu and J. Shi. "Multiclass Spectral Clustering". 2003.

Vita

Byung-Won On

Byung-Won On was born in Kumsan, South Chungcheong, Korea on Octobor 11, 1972. He received his B.E. degree in Computer Science and Engineering from Anyang University in 1998 and his M.S. degree in Computer Science and Engineering from Korea University in 2000. During his M.S. study his major research interest was in the area of *Wireless Reliable Multicast Protocols*. After he finished his M.S. study, he joined Research Center, Gluesys co., ltd., Seoul, Korea. For two years, he had worked as Senior Research Mananger in designing and developing Data Storage products (e.g., RAID Subsystems and Network Attached Storage Servers). In 2002 fall, he enrolled in the Ph.D. program in Computer Science and Engineering at the Pennsylvania State University. During his Ph.D. study, he researched main issues of Database and Data Mining, under the supervision of Dr. Dongwon Lee, the College of Information Sciences and Technology, the Pennsylvania State University. His main research interest includes Entity Resolution Problem. Furthermore, he is interested in Advanced Text Analysis (e.g., Group Join Problem, Entity Resolution Problem, Data Warehousing, and Schema Mapping), Advanced Clustering Algorithms (e.g., Web People Name Disambiguation, Parallel Clustering, and Indexing/Mining from Images and Moving Pictures), Information Fusion (e.g., Ontology-based Interoperability, Semantic Web Services Discovery and Composition, and Semantic Web), System Analysis via Social and Complex Network, Information and Knowledge Extraction, Pattern Classification, Analyzing Data with Missing Values in Bio Medical Informatics, and Information Integration and Security, and Location-based Services. In general, his research uses methods and techniques drawn from Data Mining, Numerical Linear Algebra, Theory of Graph and Networks, Complex Network in System Biology, Algebraic Graph Theory, Concurrent Matrix Computation, Algorithm Design and Analysis, Pattern Recognition, Database Systems, Applied Statistics, Stochastic Process, Neural Networks, Computer Architecture, Computer Networks, Fault Tolerant Systems, and Reliable Data Communication.