The Pennsylvania State University

The Graduate School

EFFECTIVE SOLUTIONS FOR NAME LINKAGE AND THEIR APPLICATIONS

A Thesis in

Computer Science and Engineering

by

Ergin Elmacioglu

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

August 2008

The thesis of Ergin Elmacioglu was reviewed and approved* by the following:

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

Piotr Berman
Associate Professor of Computer Science and Engineering

Wang-Chien Lee
Associate Professor of Computer Science and Engineering

Patrick Reed
Associate Professor of Civil Engineering

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

In order to identify entities, their *names* (e.g., the names of persons or movies) are among the most commonly chosen identifiers. However, since names are often ambiguous and not unique, confusion inevitably occurs. In particular, when a variety of names are used for the same real-world entity, detecting all variants and consolidating them into a single canonical entity is a significant problem. This problem has been known as the *record linkage* or *entity resolution* problem. In order to solve this problem effectively, we first propose a novel approach that advocates the use of the Web as the source of collective knowledge, especially for cases where current approaches fail due to incompleteness of data. Secondly, we attempt to mine semantic knowledge hidden in the entity context and use it with the existing approaches to improve the performance further. Finally, we the mixed type of linkage problem where contents of different entities are mixed in the same pool. Our goal is to group different contents into different clusters by focusing on extraction of the most relevant input pieces. We also illustrate the use of the proposed techniques in various real world applications.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am most grateful to my advisor, Dongwon Lee, for the guidance, patience, encouragement and support he has shown me during my time in graduate school. I thank my committee members, Piotr Berman, Wang-Chien Lee, Patrick Reed, and Reka Albert, for their insightful commentary on my work.

Finally, I want to acknowledge all my professors and colleagues who are not mentioned above. Their encouragements and advice are really appreciated.

Chapter 1

Introduction

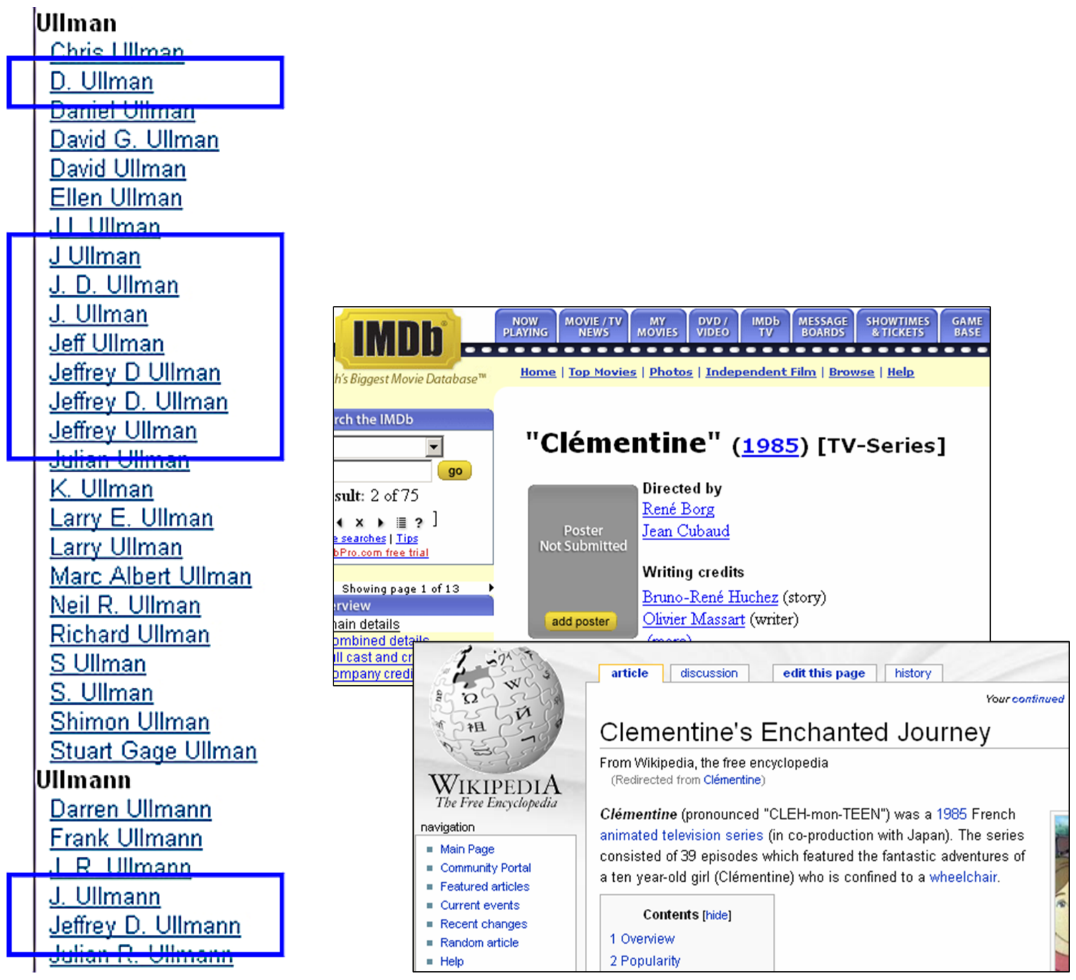
Databases, digital libraries, and data repositories are often afflicted with duplicate entities. A duplicate entry is an entry that has different identifiers that refer to the same real-world object. The problem often stems from the fact that the chosen identifier is not unique. It reflects a bad choice of primary keys in database jargon. A good example is the scenario in which *names* are used as identifiers. Names of person, movie, or organization are frequently used as unique identifiers because of their simplicity and intuitive association. However, due to various reasons, such as errors introduced during data entry or errors caused by imperfect data collection software, a variety of names may be used in reference to the same entity. One example of the problem is that an author, "John Doe," tends to fully spell in many of his publications, but may prefer the shorter "J. Doe" on others.

Among duplicate entities, let us denote the single authoritative entry as the canonical entity, while the rest will be duplicate entities or duplicates for short. Such a problem is, in general, known as the *(record) linkage* or *entity resolution* (ER) problem. The linkage problem occurs frequently in many applications and is especially exacerbated when data are integrated from heterogeneous sources.

Another important problem that can arise is confusion due to homonyms. If two persons spell their names identically (e.g., “John Doe”), how can we distinguish them? This problem is often referred to as the *name disambiguation* problem. In this type of the problem, when more than one entity with the same name representation is merged into that single name, confusion may occur.

These problems occur in many real world applications (see Motivation) and greatly affect the quality of data repositories and, therefore, user experience. Identifying all duplicate names in a collection and consolidating them into one canonical name, or correctly grouping elements of disambiguated names into individual entries, have a direct impact in many applications. Note that this ambiguous name problem cannot be completely avoided because not all entities in data collections carry a standard based unique ID system, such as digital object ID (DOI) [71].

In this thesis, we tackle the (record) linkage problem of named entities. We separate the problem into two main cases - **Split Name Linkage** and **Mixed Name Linkage**. In the split name linkage problem, an entity and its elements may be listed under more than one name (variants). The goal is to identify all of the name variants, collapse them and put all of the elements under a single, canonical name. In the mixed name linkage problem, on the other hand, elements of more than one entity that are spelled identically may be mixed together under a single name. Then the goal is to be able to identify the subgroups of the single group, each of which corresponds to the group of elements of a different entity with that name. While we have also proposed a solution for the mixed name linkage problem, our main focus in this thesis is the split name linkage problem.



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

"Clémentine" (1985) [TV-Series]

Directed by
René Borg
Jean Cubaud

Writing credits
Bruno-René Huchez (story)
Olivier Massart (writer)

Clementine's Enchanted Journey

From Wikipedia, the free encyclopedia
(Redirected from Clémentine)

Clémentine (pronounced "CLEH-mon-TEEN") was a 1985 French animated television series (in co-production with Japan). The series consisted of 39 episodes which featured the fantastic adventures of a ten year-old girl (Clémentine) who is confined to a wheelchair.

Contents [hide]
1 Overview
2 Popularity

(a) ACM (b) IMDB & Wikipedia

Figure 1.1. Duplicate names: (a) ten duplicate entities of the author "J. D. Ullman" in the ACM digital library, and (b) two duplicate titles of the same movie in IMDB and Wikipedia.

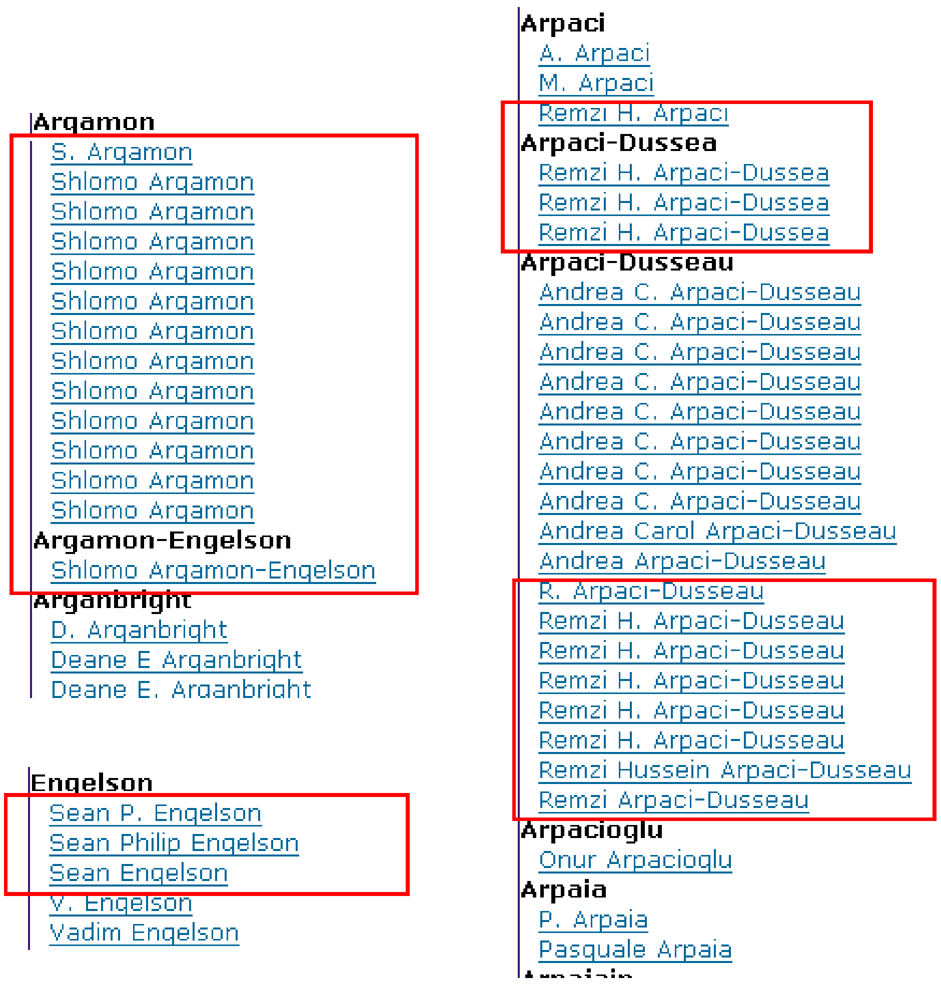
1.1 Motivation

In order to demonstrate the need for effective solutions for the name linkage problem, let us consider the following example:

In the ACM digital library (ACM-DL) [1], the full name of an author is used as an identifier (Figure 1.1(a)). Therefore, in order to access citations of the author, "J. D. Ullman', one must look for his name. However, since the ACM digital library was constructed by software that extracts metadata (e.g., author, title) from articles with little human intervention, it tends to contain errors. Therefore, citations of the author, “J. D. Ullman”, are scattered throughout the ACM digital library under ten duplicate names, thereby, causing confusion. Similarly, Figure 1.1(b) shows the example in which two repositories use different titles (i.e., names) of the same movie, “Clementine”. This would cause the linkage process to be erroneous, if combining records from these two heterogeneous resources.

Such examples of duplicate entities on the Web are common. For instance, suppose that a person, “John Doe,” moves from organization x to organization y and appears on a different home page at each organization, one using “John Doe” and the other using “J. Doe”. Being able to identify whether or not the two home pages refer to the same person is an interesting challenge. A person’s name may also change, partially or entirely, for various reasons, such as marriage, divorce, etc. Some real world examples from the ACM-DL appear in Figure 1.2. The right side of Figure 1.2 illustrates a case where the author “Remzi H. Arpaci” is also listed under a name with an addition to the last name (“Remzi H. Arpaci-Dusseau”). The left side shows a more challenging case where the author “Sean P. Engelson” has also entries under the name completely changed to “Shlomo Argamon”.

The split name problem is not limited to persons’ names. Any real-world entity may have multiple names. For instance, the mp3 player, “*Apple iPod Nano 4GB*”, may be listed with this product title on the manufacturer’s web site, although it may appear under a slightly different title, “*Apple iPod Nano 4GB*”, on another e-commerce site. In addition, due to various reasons (e.g., marketing strategies), different products may be sold under different names in different regions of the world. One example is a five-door hatchback automobile that was introduced in



Argamon
S. Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Shlomo Argamon
Argamon-Engelson
Shlomo Argamon-Engelson
Arganbright
D. Arganbright
Deane E Arganbright
Deane E. Arganbright

Engelson
Sean P. Engelson
Sean Philip Engelson
Sean Engelson
V. Engelson
Vadim Engelson

(a)

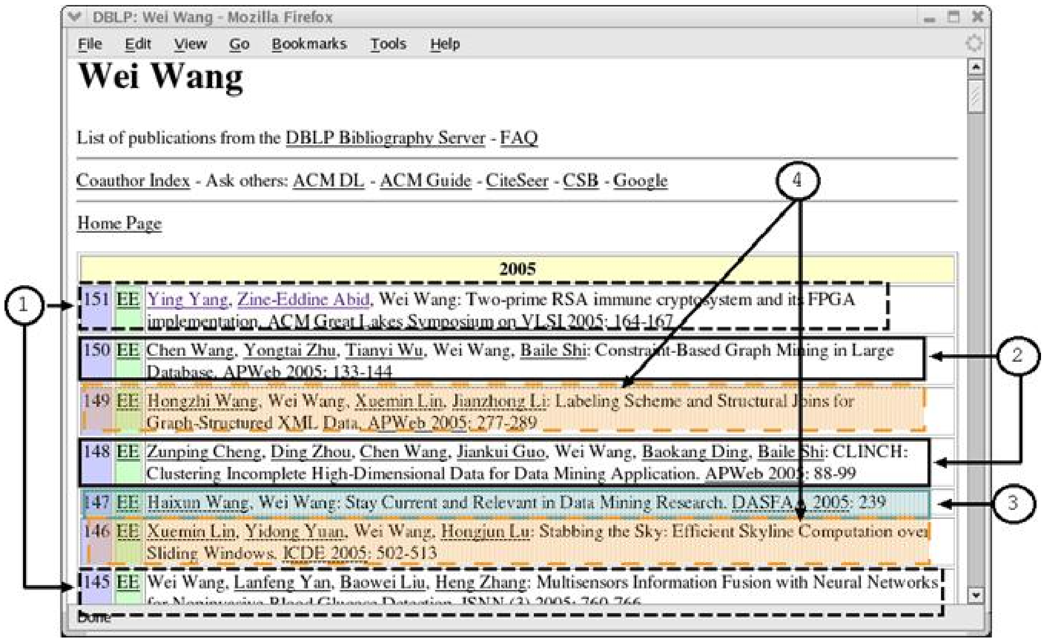
Arpaci
A. Arpaci
M. Arpaci
Remzi H. Arpaci
Arpaci-Dussea
Remzi H. Arpaci-Dussea
Remzi H. Arpaci-Dussea
Remzi H. Arpaci-Dussea
Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
Andrea Carol Arpaci-Dusseau
Andrea Arpaci-Dusseau
R. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau
Remzi Hussein Arpaci-Dusseau
Remzi Arpaci-Dusseau
Arpacioglu
Onur Arpacioglu
Arpaia
P. Arpaia
Pasquale Arpaia

(b)

Figure 1.2. Duplicate entities in the ACM-DL due to partial or full name changes: (a) Sean Engelson vs. Shlomo Argamon and (b) Remzi H. Arpaci vs. Remzi H. Arpaci-Dusseau.

June 2001 by Honda. The vehicle is known as the *Fit* in Japan, China, and North and South America. However, it is called as the *Jazz* in Europe, the Middle East, South East Asia and Africa. Furthermore, even companies may use different names. For example, *Tefal* is a French cookware and small appliance manufacturer. However, its products are sold under the name, *T-Fal*, in North America, Brazil and Japan.

The mixed name problem also occurs frequently in many applications. Fig-


DBLP: Wei Wang - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
1
2
3
4
Done

Wei Wang

List of publications from the DBLP Bibliography Server - FAQ

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

Home Page

2005		
151	EE	Ying Yang, Zine-Eddine Abid, Wei Wang: Two-prime RSA immune cryptosystem and its FPGA implementation. ACM Great Lakes Symposium on VLSI 2005: 164-167
150	EE	Chen Wang, Yongtai Zhu, Tianyi Wu, Wei Wang, Baile Shi: Constraint-Based Graph Mining in Large Database. APWeb 2005: 133-144
149	EE	Hongzhi Wang, Wei Wang, Xuemin Lin, Jianzhong Li: Labeling Scheme and Structural Joins for Graph-Structured XML Data. APWeb 2005: 277-289
148	EE	Zunping Cheng, Ding Zhou, Chen Wang, Jiankui Guo, Wei Wang, Baokang Ding, Baile Shi: CLINCH: Clustering Incomplete High-Dimensional Data for Data Mining Application. APWeb 2005: 88-99
147	EE	Haixun Wang, Wei Wang: Stay Current and Relevant in Data Mining Research. DASFAA 2005: 239
146	EE	Xuemin Lin, Yidong Yuan, Wei Wang, Hongjun Lu: Stabbing the Sky: Efficient Skyline Computation over Sliding Windows. ICDE 2005: 502-513
145	EE	Wei Wang, Lanfeng Yan, Baowei Liu, Heng Zhang: Multisensors Information Fusion with Neural Networks for Noninvasive Blood Glucose Detection. ISNN (3) 2005: 760-766

Figure 1.3. Mixed names: at least four different "Wei Wang"'s citations are mixed in DBLP-DL [64].

ure 1.3 shows one example taken from the DBLP digital library (DBLP-DL) [24]. All the citations of the name, "Wei Wang," are listed under this single name. However, when the citations are analyzed, it is believed that there are at least four different scientists who have this citations mixed under the same name spelling.

A web search based on a person's name has always been one of the most popular query types. For instance, a recent study in [36] reports that around 30% of search engine queries include a person's name. However, persons' names are inherently ambiguous (e.g., 90,000 distinct names are shared by 100 million people in the US) [36]. Therefore, when a user searches a person's name on the web, the top-ranked search results are likely to include a mix of web pages links that belong to multiple individuals under the same name spelling. Table 1.1 illustrates the situation for multiple cases. It shows the number of different entities available in the top 98-100 web search results of queries for seven person' names from the English

Table 1.1. Name disambiguation on the Web Search [7]

Name	**entities**	**documents**
	Wikipedia names	
John Kennedy	27	99
George Clinton	27	99
Michael Howard	32	99
Paul Collins	37	98
Tony Abbott	7	98
Alexander Macomb	21	100
David Lodge	11	100
Average	23,14	99,00
	ECDL-06 Names	
Edward Fox	16	100
Allan Hanbury	2	100
Donna Harman	7	98
Andrew Powell	19	98
Gregory Crane	4	99
Jane Hunter	15	99
Paul Clough	14	100
Thomas Baker	60	100
Christine Borgman	7	99
Anita Coleman	9	99
Average	15,30	99,20

Wikipedia [86], and 10 persons' names from the program committee member list of the European Conference on Digital Libraries (ECDL) in 2006. The Wikipedia names are popular or historical names. Hence these names are expected to have a few number of predominant entities in the top ranked search results [7]. Yet, we see as many as 37 different entities mixed for the name, "Paul Collins", in the top 98 search engine results. In general, the ECDL names are less ambiguous. Yet, if one searches for "Thomas Baker", he finds a mixed list of links to the web pages of 60 different persons called "Thomas Baker" in the top 100 results. This makes it very difficult to decide which link(s) belongs to the person whom the user is searching for.

1.2 Problem Definition

We now formally define the name linkage problem. To make the presentation simple, let us assume that our problem setting has one *canonical entity* and the goal is to find all of its duplicate entities. However, note that subsequent discussion of our proposal is orthogonal to the issue of canonical entity since it works as long as one of entities is designated as the canonical one. Therefore, the issue of determining the *canonical entity* among many candidate entities is not pursued further in this work. Formally, the general name linkage problem in our setting is defined as follows:

Definition Given a set of entities, E, where each entity e_i has a name and information $I(e_i)$ (e.g., contents, affiliation, social network knowledge), for each canonical entity, e_c ($\in E$), identify all duplicate entities, e_v ($\in E$) by using a distance function (based on the Web, entity context graph etc.), such that $dist(I(e_c), I(e_v)) < \phi$, where ϕ is a pre-set threshold.

A naive algorithm with $O(|E|^2)$ running time for the general name linkage problem is:

```
for each entity $e_c \in E$
    for each entity $e_v \in E$, where $e_c \neq e_v$
        if $dist(I(e_c), I(e_v)) < \phi$, then $e_v \sim e_c$
```

Note that our main focus in this thesis is *not* the algorithmic improvement of the general linkage problem. Rather, we study how to devise better $dist()$ functions (web based, graph based etc.) for novel and effective solutions towards the name linkage problem.

1.3 Thesis Overview

This thesis is organized as follows. In the following chapter, we review the background information and present the related work for both *Split Name Linkage* and *Mixed Name Linkage* problems.

In chapter 3, we tackle the problem of *Split Name Linkage*. We advocate the use of the Web as the collective knowledge of the people. As the largest source of information, the Web usually provides a good representation of what people think. We show how to apply it to the changeling cases of the name linkage problem for where traditional approaches fail due to incompleteness of data. We also present an application of our method to link short-forms to long-form names.

In chapter 4, we improve the performance of traditional approaches by mining the hidden relations in the context of the entities to be linked. Although functioning in many scenarios, traditional approaches often suffer from a large number of false positives. To cope with this problem, we propose a framework to represent entities' names and associated content on a graph and extract the semantic information to improve the precision of the standard distant metrics. We first study how the social network can be generated and what characteristics are available for the linkage process. Then we propose our precision-aware linkage method based on the notion of quasi-cliques on social networks. Finally, we study how to improve the data quality on the social network by a network model.

In chapter 5, we tackle the problem of the Mixed Name Linkage, particularly on the Web. We propose several methods to extract and utilize various types of information from the data to be clustered and provide the processed input to any clustering algorithm for a more robust linkage of the mixed names of different entities.

Finally, we conclude by summarizing our methods and contributions. We also outline future work in chapter 6

Chapter 2

Background

Overall, the linkage problem has been known as various names – record linkage (e.g., [30, 16]), citation matching (e.g., [57]), identity uncertainty (e.g., [67]), merge-purge (e.g., [40]), object matching (e.g., [19]), entity resolution (e.g., [74, 5]), authority control (e.g., [83, 43]), and approximate string join (e.g., [35]) etc.

The two types of the linkage problem (split vs. mixed) have first been defined by [64] as entity resolution problems. Although the proposed methods in [64] are also applicable for our setting, our main focus is to solve the linkage problem for entities with names. In this thesis, we propose novel and effective solutions particularly for the linkage of *named* entities.

2.1 The Split Name Linkage Problem

Bilenko et al. [16] have studied name matching for information integration using string-based and token-based methods. Cohen et al. [22] have also compared the efficacy of string-distance metrics, like JaroWinkler, for the name matching task. [50] experimented with various distance-based algorithms for citation matching, with a conclusion that word based matching performs well.

On et al. [63] conducted an in-depth study on the split entities case from the blocking point of view.

Unlike the traditional methods exploiting textual similarity, Constraint-Based Entity Matching (CME) [76] examines “semantic constraints” in an unsupervised way. They use two popular data mining techniques, Expectation-Maximization (EM) and relaxation labeling for exploiting the constraints. [14] presents a generic framework, *Swoosh* algorithms, for the entity resolution problem. The recent work by [25] proposes an iterative linkage solution for complex personal information management. Their work reports good performance for its unique framework where different linkage results mutually reinforce each other (e.g., the resolved co-author names are used in resolving venue names).

The recent trend in the linkage problem shows similar direction to ours (e.g., [16, 15, 53, 48]). Although each work calls its proposal under different names, by and large, most are trying to “exploit additional information beyond string comparison.” For instance, [48] 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. A more extensive and systematic study is needed to investigate the usefulness and limitations of the context in a multitude of the linkage problem. The main difference from ours to these approaches lies in the simplicity of our idea of using the Web to extract additional information.

Finally, to overcome the data incompleteness problem, several studies using the Web through search engines are also available. Among those, [81] uses the statistical data by querying the Web for recognizing synonyms, and in [55], syntactic patterns are searched in the Web by using the Google API in order to acquire background knowledge for anaphora resolution. In addition, in [2], related texts are crawled from the Web to enrich a given ontology. In [21, 34, 72], other approaches are employed to find the best concept for an unknown instance in a given ontology. [56] uses page counts to determine the strength of relations in a social network. A formal study by [20] defines a semantic distance metric between two terms using page counts from the Web. In [78], authors tackle another type of entity resolution problem known as “mixed citation problem” [51] using URL information. Our web-based linkage method is the first study as to utilize the collective knowledge of the Web through a search engine for the split name linkage problem.

2.2 The Mixed Name Linkage Problem

In general, the solutions for the mixed name linkage problem can be categorized as clustering (i.e., [39], [13]) or classifications (i.e., [38]) approaches.

Han et al. [38] propose two supervised learning-based approaches for citation data sets using the Naive Bayes model and the Support Vector Machines [82, 23]. Furthermore, Han et al. [39] present an unsupervised learning approach using K-way spectral clustering, again for citation data sets.

In addition, Malin [53] utilizes hierarchical clustering into unsupervised name disambiguation, relying upon exact name similarity. In this same study, Malin also employs another method based on random walks between ambiguous names on a global social network constructed from all sources. Lee et al. [51] propose a citation labeling algorithm, considering the problem for the scalability perspective. Recently Bekkerman et al. [13] propose techniques to disambiguating collections of Web appearances using Agglomerative and Conglomerative Double Clustering.

In [78], authors also use hierarchical clustering for the mixed citation problem by leveraging the collective information on the web to do disambiguation, by employing a web search engine.

Chapter 3

The Split Name Linkage Problem

In this chapter, we focus on the split name linkage problem in particular cases where current information associated to names has not enough content or is not discriminative enough to link the variant names. For such cases, we attempt to acquire the outside knowledge and consult people what they think. If many people collectively agree that two entities are the same, then two entities are probably the same. Furthermore, we argue that the Web is a good representation of what people think. That is, we propose to seek for additional information of an entity from the Web. Our hypothesis is the following:

If an entity e_1 is a duplicate of another entity e_2, and if e_1 frequently appears together with information I on the Web, then e_2 may appear frequently with I on the Web, too.

Therefore, by measuring how the information I appears together with e_2 on the Web, we can determine if e_2 is a duplicate of e_1 or not. In particular, we propose various methods to capture the information I of an entity from the Web. Since our methods rely on search engines such as Google or MSN to draw additional information of an entity, compared to a baseline method, ours is simpler and more

Table 3.1. Notations used.

Notation	Description
e_1, e_2, ..	an entity
e_c	a canonical entity
E	an entity set
t_1, t_2 ..	token
tf	term frequency
*tf*idf*	term freq.* inverse document freq.
*w_tf*idf*	web information based $tf * idf$
$name(e_i)$	name of entity e_i
$token(e_i)$	all tokens of entity e_i's content
$data(e_i)$	representative data piece of entity e_i's content
$count(x)$	number of pages returned by query string "x"
$host_k(x)$	top-k host names returned by query string "x"
$x \cap y$	equivalent to "x `AND` y" in a query string
$x \cup y$	equivalent to "x `OR` y" in a query string

practical.

Our contributions in this work are as follows:

- To solve the linkage problem, we propose a novel idea of using the *Web* as a source for additional knowledge of an entity. Since the Web is the largest source of knowledge on any kind of information, acquiring additional information of an entity becomes possible, regardless of domains or applications.

- To capture the additional information of an entity returned from a search engine, we propose various methods such as using the frequency of representative terms, URL information or the content of returned web pages.

- Through an array of experimentation, we validate our hypothesis and demonstrate the feasibility of the proposed methods. Our proposal shows 51% (on average) and 193% (at best) improvement in precision/recall compared to a baseline approach.

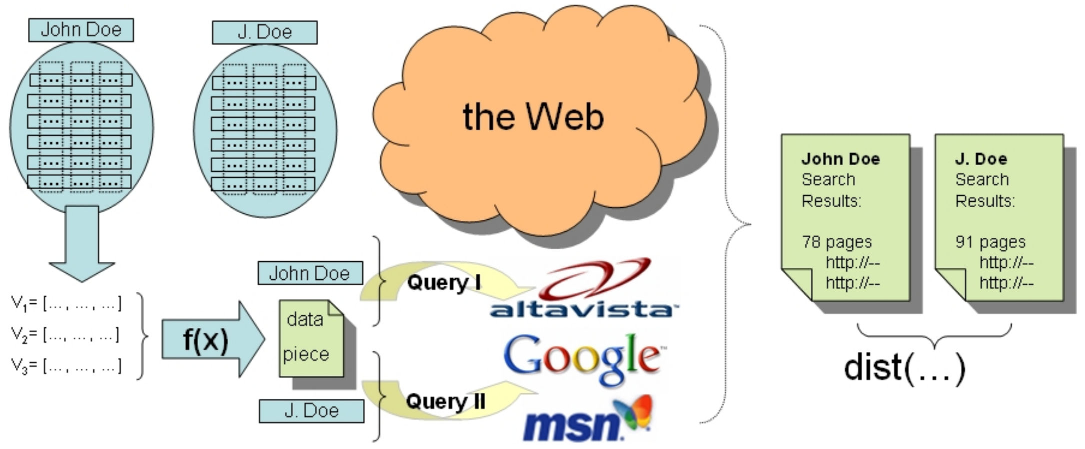


Figure 3.1. The overview of the web-based name linkage.

3.1 Solution Overview

Throughout the chapter, we use notations in Table 3.1.

The basic framework of our approach is illustrated in Figure 3.1. Consider a canonical entity e_c and a candidate entity e_1 that is potentially a duplicate of e_c. Name descriptions of entities are available as $name(e_c)$ and $name(e_1)$. Then, our approach consists of three steps as follows:

1. (Section 3.2.1) We identify the best representative data piece (or pieces) of e_c. This data piece could be anything – from as simple as a single token of e_c to tuple(s) or as long as the entire contents of e_c. Suppose we get such a data piece from e_c, called $data(e_c)$.

2. (Section 3.2.2) We acquire a collective knowledge of people to see how "often" $data(e_c)$ is used together with e_1. For instance, we submit two queries to a search engine: Q_1:"$name(e_c)$ `AND` $data(e_c)$" and Q_2:"$name(e_1)$ `AND` $data(e_c)$". To these queries, search engines would return web pages that contain both keywords of a query.

3. (Section 3.2.3) By analyzing the returned pages in various aspects (e.g., frequency, URLs, documents), then we measure how similar e_1 is to e_c.

Note the existence of abundant avenues for variations here. For instance, we may change the way we submit two keywords in a query such as "*name* `OR` *data*" or "*name* `NOT` *data*". In addition, in analyzing the returned pages, we may use the number of total match, contents of top-k returned pages, host names of URLs, etc. In the following, we investigate such variations to determine their effectiveness to the linkage problem.

3.2 Web-based Split Name Linkage

3.2.1 Representative Data of an Entity

Suppose one wants to determine if “G. Bush” is a duplicate of “V. Bush” or not. Depending on the context, this task may or may not be easy to resolve. However, if one knows that the representative data of “G. Bush” is, say, “White House”, then one may measure how often “V. Bush” is used together with “White House” on the Web, and conclude that “V. Bush” is not a duplicate entity of “G. Bush”. Therefore, the first step of our proposal is to identify the representative data of an entity e, denoted as $data(e)$.

Ideally, $data(e)$ helps discriminate duplicates of e. If $data(e)$ itself occurs too often on the Web, then it does not help link duplicates. On the other hand, if $data(e)$ itself occurs too rarely on the Web, then its discriminating power gets too weak to be used, and can be easily influenced by noises.

To determine if an entity e_1:“Jeffrey D. Ullman” is the duplicate of an entity e_2:“Ullman, S.”, using “aho” (the last name of one of the collaborators) as a representative data of e_1 may yield a good result. However, using “professor” as the representative data of e_1 may not be a good choice since Google returns the CS Professor “Shimon Ullman” in Israel higher (also more frequent) than the CS Professor “Jeffrey D. Ullman” in US.

By adopting conventional approaches in Information Retrieval, we propose to use three measures to select the best representative token of an entity as follows:

- *tf*: One may assume that the more frequently a token appears in an entity, the better representative the token gets. Therefore, we may use the traditional *term frequency (tf)* as a metric[1], and use the token that occurs most frequently as the representative.

- *tf*idf*: When the token selected by *tf* too common like "computer" or "web", its discriminating power diminishes. Even if we get relevant web pages for an entity e, such a common token may also frequently appear with many other candidate entities, and thus may not help find the actual duplicates. Hence the popular *term frequency * inverse document frequency (tf*idf)* [73] scheme is more appealing since it can find more discriminative tokens in an entity's data contents[2].

- *w_tf*idf*: Finally, we propose a web based scheme called *w_tf*idf*, which resembles *tf*idf* but gets the frequency information from the Web. It is defined as:

$$w\_tf * idf(e, t_i) = \frac{count(name(e) \cap t_i)}{count(t_i)}$$

  where e is an entity, t_i is a token from e's contents, and $count(x)$ is a function returning the number of occurrences of the term x on the Web. That is, *w_tf*idf* is based on the observation that the more t_i appears with e and the less t_i independently appears on the Web, the more valuable t_i gets to discriminate e.

[1]From the corpora (i.e., contents or metadata of entities), we first remove all stop-words (e.g., "a" or "the").

[2]Even further, probabilistic topic models such as LDA [17] can be used on the content to discover a pre-specified number of latent topics, each of which has a particular probability of emitting a specific token. Then one or more of such topics could be utilized as a representative data for the entity in question. We leave this exploration as future work.

3.2.2 Knowledge Acquisition from the Web

Once a representative token t_c is selected for a canonical entity e_c, suppose one wants to determine if another candidate entity e_i is a duplicate entity of e_c or not. Then, one forms two queries Q_1 and Q_2 and submit to the Web:

$$Q_1\text{: “}\textit{name(}e_c\textit{)}\ \texttt{AND}\ t_c\text{” and } Q_2\text{: “}\textit{name(}e_i\textit{)}\ \texttt{AND}\ t_c\text{”}$$

In return, one receives two separate answer collections (for Q_1 and Q_2) of relevant web pages that contain *both* entity names and the representative token in question. Note that the potential downside of acquiring knowledge from the Web is the degraded performance – i.e., each request to search engines involves expensive network traffic and web page pre- and post-processing. However, we argue that the proposed web-based name linkage technique is *not* to replace existing linkage solutions that are fast and accurate when there are enough information. Rather, ours aims at situations when the linkage problem cannot be solved well due to the lack of information. In such scenarios, we believe that one can trade the speed for the improved accuracy. Nevertheless, we study the issue of scalability, and propose an alternative that uses a local snapshot of the Web, instead of search engines, to improve the speed (in Section 3.2.4.3).

3.2.3 Interpreting the Collective Knowledge

To the aforementioned queries Q_1 and Q_2, search engines return abundant results. Therefore, next step is to analyze the returned results to unearth the collective knowledge of the Web. We mainly propose three methods[3] as follows:

[3] In [78, 28], we have reported a promising result using page count and URL in both name disambiguation and linkage problems. We greatly expand them in this work.

Table 3.2. Heuristics to create a virtual document v from top-k returned web pages.

Notation	Description
$D(m)$	Top m ($\leq k$) documents are concatenated
$D(sim, m)$	m of the top-k web pages where the contents have the highest similarity with the current contents of the entity in question (i.e., publication information available in the data set)
$T(1, n)$	Top n ($\in \{50, 100, 200, 1000\}$) tokens with the highest *tf*idf* weight from the top ranked web page
$T(all, n)$	Top n ($\in \{100, 200, 400, 1000, 10000\}$) tokens with the highest *tf*idf* weight from all top-k web pages
$T(each, n)$	Top n ($\in \{10, 20, 40, 100\}$) tokens with the highest *tf*idf* weight from each of top-k web pages
$T(common, p)$	Common tokens in at least p ($\in \{2, 3, 4\}$) documents from top-k web pages
$T(common, p, n)$	Common tokens in the top-n ($\in \{10, 20, 50\}$) tokens with the highest *tf*idf* weight of at least p ($\in \{2, 3, 4\}$) documents from top-k web pages
$S(m, n)$	A set of sentences of the top-m ranked web pages such that the sentence contains one of the top-n ($\in \{100, 200\}$) tokens with the highest *tf*idf* weight
$Snippet(m)$	Snippets of top-m ($\leq k$) web pages (i.e., short summary-like information provided by search engines) are concatenated

(1) ***Page Count***. Search engines return the number of web pages that satisfy the constraints of the given query string among entire indexed pages. Suppose an entity e_2 is a duplicate of an entity e_1, but e_3 is not. Further, a token t_i is selected as the representative of e_1. Then, it is plausible that the appearance of $e_1 \cap t_i$ and $e_2 \cap t_i$ on the Web will be *relatively* similar than that of $e_1 \cap t_i$ and $e_3 \cap t_i$. That is, $|count(e_1 \cap t_i) - count(e_2 \cap t_i)| \ll |count(e_1 \cap t_i) - count(e_3 \cap t_i)|$. Formally, the similarity between two entities e_c and e_i is defined as: (1) if $count(name(e_c) \cap t_c) = count(name(e_i) \cap t_c)$, then $sim(e_c, e_i) = 1$, and (2) otherwise, $sim(e_c, e_i) = \frac{1}{|count(name(e_c) \cap t_c) - count(name(e_i) \cap t_c)|}$, where e_c and e_i are canonical and candidate entity, respectively, and t_c is the best representative token of e_c.

(2) ***URL***. From top-k results return from search engine, we extract their host addresses from the URLs, and use their overlap like Jaccard similarity as the similarity: $sim(e_c, e_i) = \frac{|host_k(name(e_c) \cap t_c) \bigcap host_k(name(e_i) \cap t_c)|}{|host_k(name(e_c) \cap t_c) \bigcup host_k(name(e_i) \cap t_c)|}$, where $host_k(x)$ is the collection of host names of top-k URLs returned for the search query x.

(3) ***Web Pages***. If one views the results of Q_1 and Q_2 as *virtual documents*,

then their similarity can be computed using document to document similarity measures. Then, depending on how to create a virtual document among a set of returned web pages, a variety of alternatives can be used. Table 3.2 lists a set of heuristics that we used to create a virtual document. Suppose we created two virtual documents, D_i and D_j, using one of approaches in Table 3.2. Then, we propose to use either jaccard, cosine, or language model based similarity. The jaccard similarity of D_i and D_j is the ratio between intersected vs. unioned token sets of two documents: $sim_{jaccard}(D_i, D_j) = \frac{|tokens(D_i) \bigcap tokens(D_j)|}{|tokens(D_i) \bigcup tokens(D_j)|}$, and the cosine similarity of D_i and D_j uses the cosine of the angle between two m-dimensional vectors where m is the number of distinct tokens in corpora) as the similarity: $sim_{cosine}(D_i, D_j) = \frac{\sum_{k=1}^{n} token_k(D_i) token_k(D_j)}{\|D_i\| \|D_j\|}$. For further details of these similarities, refer to [22].

Probabilistic language models have been successfully applied to the ad-hoc IR tasks (e.g., [87]). In the language model approach, each document is represented by a unigram word distribution θ_d. Then, the *Kullback-Leibler divergence* can be used to measure how the language models of two documents differ:

$$KL(\theta_i \| \theta_j) = \sum_k p(k|\theta_i) log \frac{p(k|\theta_i)}{p(k|\theta_j)}$$

Here, θ_d can be found by the *maximum likelihood estimation* (MLE): $P(token_i|D) = \frac{tf(token_i, D)}{\sum_{token_j} tf(token_j, D)}$. In [87], authors use the KL divergence on document models to detect the novelty of relevant documents in an adaptive filtering system. For our language model based measures, we may adopt the same approach by measuring the novelty of canonical entity's virtual document, $D_i(e_c)$, against each candidate entity's virtual document, $D_j(e_v)$. Then, the lower the novelty score of $(D_i(e_c), D_j(e_v))$ is, the higher the similarity between two entities e_c and e_v gets.

Table 3.3. Summary of data sets. Numbers in parenthesis represent average # of elements (i.e., citations, movie titles, etc.) for each entity

Data set	# of e_c	avg. # of e_v	avg. # of duplicates
ACM	43 (14.2)	21 (3.1)	1.8 (6.7)
ArXiv	64 (3.4)	9 (8.1)	1.3 (27)
IMDB	50 (24)	20 (24)	1 (23.5)

Finally, smoothing techniques are necessary to adjust the MLE to assign non-zero probabilities to unseen tokens to make KL-based measure more appropriate. We attempt to use two smoothing techniques that are also used in [87]:

- *Bayesian Smoothing using Dirichlet Priors.* This technique uses the conjugate prior for a multinomial distribution (i.e., Dirichlet distribution) with parameters: $(\lambda p(token_1), ..., \lambda p(token_n))$. Then, the model is given by

$$P_\lambda(token_i|D) = \frac{tf(token_i, D) + \lambda p(token_i)}{\sum_{token_j}(tf(token_j, D) + \lambda p(token_j))}$$

  In consistent with [87], in our experiments, if $token_i \in D$, then we set $\lambda p(token_i) = 0.5$, and 0 otherwise.

- *Smoothing using Shrinkage.* This technique models each document from the language model of the document θ_{D_MLE} and a model for general English θ_{E_MLE} built from all documents in the system, by: $\theta_D = \lambda_D \theta_{D\_MLE} + \lambda_E \theta_{E\_MLE}$, where $\lambda_D + \lambda_E = 1$. We experimentally determine the best value for the parameters λ_D and λ_E.

3.2.4 Experimental Validation

In experiments, we seek to answer the following questions:

- Q1: Is Hypothesis 1 valid? That is, how do web-based name linkage schemes perform?

- Q2: Which of the proposed web-based name linkage schemes performs the best? Which variations?
- Q3: Which method to interpret the collective knowledge is better?
- Q4: How do results change for different search engines?

All experimentation was done using MySQL Server 5.0 on a desktop with AMD Opteron 2GHZ and 4GB RAM. As search engines for our methods, we used both Google and MSN.

3.2.4.1 Set-up

We used three data sets, as summarized in Table 4.3 – *ACM Digital Library* for computer science authors with her citation list, *ArXiv Digital Library* for general science authors with her citation list, and *Internet Movie Database (IMDB)* for actors with movie-related data in which she stars. In this context, the linkage problem is to identify duplicate author entities in ACM and ArXiv and actor entities in IMDB.

For each canonical entity e_c, to find its duplicates, comparing e_c to each entity e_v in E ($e_c \neq e_v$) is prohibitively expensive (recall the naive algorithm in chapter 1). Therefore, often, a pre-processing step, called a *blocking*, pre-filters entities in E into a small set of candidate entities, called a *block*. In the experiments, we used a heuristic blocking rule that showed good performance for name-related entity resolution problem in [63] – i.e., all (author or actor) entities that share the same last name as the canonical entity are clustered into the same block. Then, duplicate entities are injected into the block. This block may contain as small as a few dozen entities to as large as hundreds of entities, depending on the last name in question. Hence, the goal is that for each block, a scheme is to detect all duplicates correctly.

Case I. From the ACM data set, we have gathered 43 real cases of duplicate names[4] (thus 43 blocks). These real cases include duplicates as simple as "Dongwon Lee" vs. "D. Lee" (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 duplicates of "Jeffrey D. Ullman". For instance, in the "Dongwon Lee" (i.e., canonical entity) block, there is a duplicate entity "D. Lee" as well as hundreds of author entities with the last name "Lee". Then, we examine how well our web-based name linkage schemes are able to identify the duplicate entity "D. Lee" correctly by using the collective knowledge on the Web.

Case II. Second, we gathered 64 real cases (i.e., blocks) from ArXiv data set. One important difference from ACM case is that each entity block in ArXiv case tends to have larger contents. Hence, it is expected to be easier to link entities using any $dist()$. Therefore, to make the case more challenging, for each block, we selected an author entity with the smallest number of citations as the canonical entity. Note that since each canonical entity has a significant content overlap with its duplicate entity, simple distance metrics tend to work well although they may generate many false positives too.

Case III. As the final test case, we attempt to simulate a scenario in which entities have large associated contents but they are not discriminative enough to help the linkage. For such a scenario, we collected 50 real actor names and the titles of movies they appeared from IMDB. Since IMDB is a commercial database,

[4]All 43 cases were verified by directly asking to authors themselves or checking their home pages.

it has little errors or noise, compared to ACM and ArXiv data sets. Therefore, we created 50 test cases such that each actor (i.e., canonical entity) has exactly one "a.k.a." name (i.e., duplicate entity). Since "a.k.a." name usually does not have any contents in IMDB, we randomly split the original contents of the actor, and assign each to an actor and her "a.k.a." entity. Due to the nature of the data set, the intersection of the two halves sharing the same content might be small – i.e., movie titles of an actor do not generally have a common theme.

Evaluation Metrics. We use traditional precision/recall with the window size equal to the number of duplicates. For instance, for a canonical entity e_c, suppose that there are k duplicates in the block. When a scheme returns top-k candidate duplicates, consider that only r of them are correct duplicates and the remaining $k - r$ candidates are false positives. Then, the precision and recall becomes the identical and calculated as: $Precision = Recall = \frac{r}{k}$.

As a baseline approach for the linkage problem, we use the Jaccard similarity that measure the overlapping ratio of the contents of two entities ([63] reported that although simple, Jaccard similarity showed good performance among various standard distance functions):

$$sim_{jaccard}(e_c, e_i) = \frac{|token(e_c) \bigcap token(e_i)|}{|token(e_c) \bigcup token(e_i)|}$$

For the first two cases, our study focuses on three attributes of citations only – co-author, title, and venue – since (1) too many tokens from too many attributes slow down the process, (2) recent study [37] shows these three play a major role, and (3) other attributes have often missing values. Since ArXiv is a pre-print repository, it often does not have vene information. In the IMDB case, we only

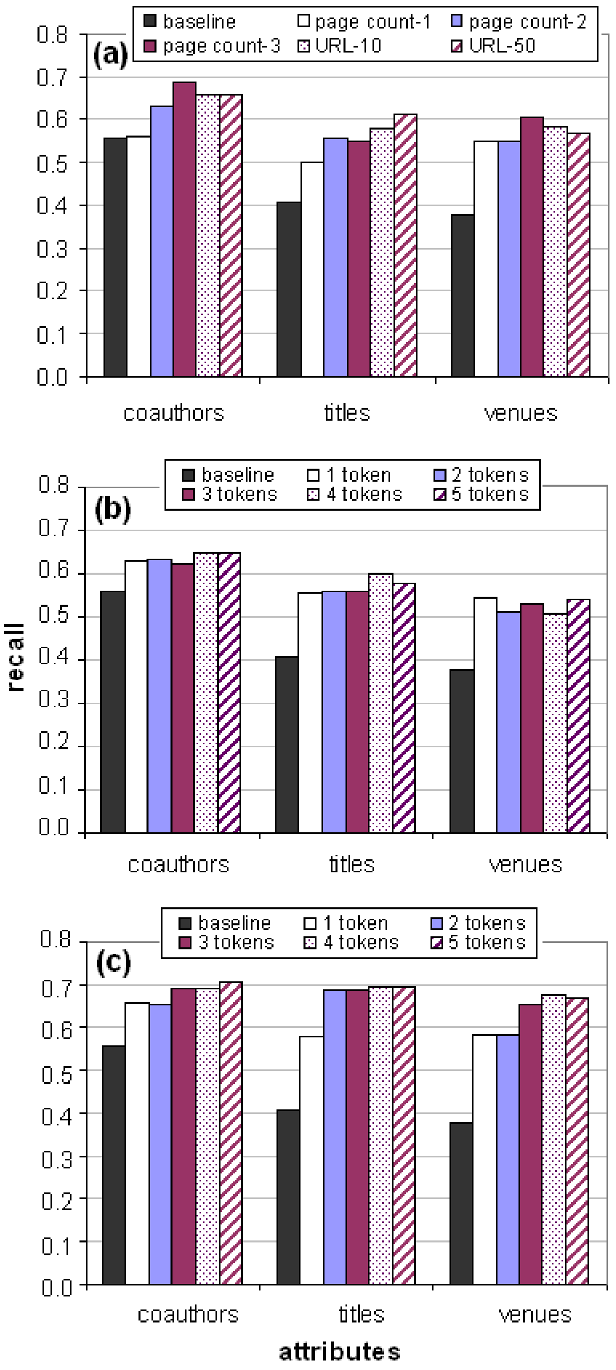


Figure 3.2. (a) ACM test case (using `Google` as search engine). Page count-1,2,3 – use *tf*, *tf*idf* and *w_tf*idf* for token selection, respectively. URL-10, 50 – use *tf*idf* for token selection and evaluate host names of top-10 and 50 URLs returned, respectively. (b, c) Effect of using multiple ($k = 2, 3, 4, 5$) tokens on Page count-2 and URL-10, respectively.

use the movie title attribute.

3.2.4.2 Accuracy

1. Page Count and URL Methods. Using the ACM data set, we first experiment with variations of the first two basic web based linkage methods: page count and URL methods. Figure 3.2(a) shows the results of these methods using `Google` against Jaccard metric on all three attributes of the ACM data set individually. All five variations of the basic methods performs better than the baseline Jaccard metric. Among three token selection schemes used by page count based methods, *w_tf*idf* yields better performance compared to the other two schemes, finding more discriminative tokens for entities. The two URL based methods show stable and near top performance on all cases probably due to the large volume of information used and the quality of the top returned documents by search engine. Figures 3.2(b,c) show the effect of using multiple tokens on the methods page count-2 and URL-10, respectively. Here, for each token selected in the top-k tokens of an entity, an experiment is run, and the similarity is calculated by averaging the individual similarities of all k experiments. The results are mostly parallel, and do not show significant improvement.

2. Web Page Based Methods. All experiments for the web page based methods were done using top-10 results from `Google` as the search engine. For the token selection method, we use *tf*idf* only, since it usually shows better discriminating power than *tf*, and *w_tf*idf* adds up extra time cost although it may have a better performance [28]. Web page based methods are based on the creation of a virtual document from the web page results for each entity and then comparing these virtual documents to make a decision for the linkage. To create a virtual document,

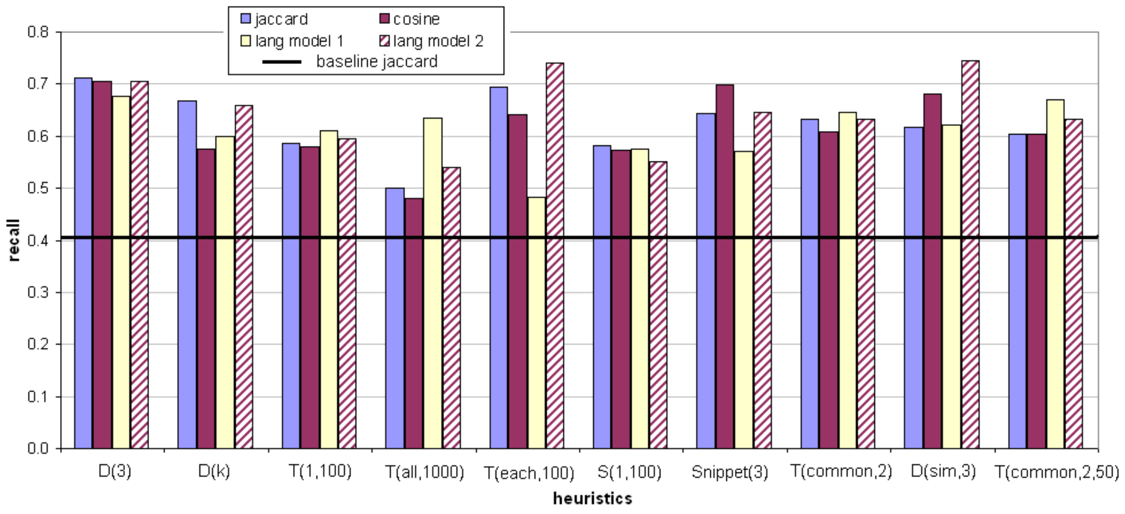


Figure 3.3. Results of the ten variations from all web page based heuristics. The experiments are on the title attribute of the ACM data set and use *tf*idf* scheme for token selection and `Google` as search engine.

we defined a number of heuristics and their variations as listed in Table 3.2. Due to space constraints, we will only show a few variations for each experiment.

To determine which one to use, Figure 3.3 shows the results of 10 variations of our web page based heuristics on the title attribute of the ACM data set. In Figures 3.3-3.6, note that the *baseline jaccard* (i.e., straight line) is the overlapping ratio of the “contents” of two entities while the plain *jaccard* is the overlapping ratio of the “virtual documents” of two entities. In our extensive experimentation, for the majority of cases, we saw that $top - 3$ returned pages are usually the most relevant ones and effective if used in virtual document creation. In other cases, using as many as returned web pages helped to achieve better precision/recall. Therefore, among the various heuristics, we selected *D(3)*, *Snippet(3)*, *D(sim,3)* due to their good performance. We also included one more variation of the first heuristic, *D(k)*, to illustrate the performance when all returned documents are used.

Figures 3.3 and 3.4(a)(b) show the ACM results on the title, coauthor, and

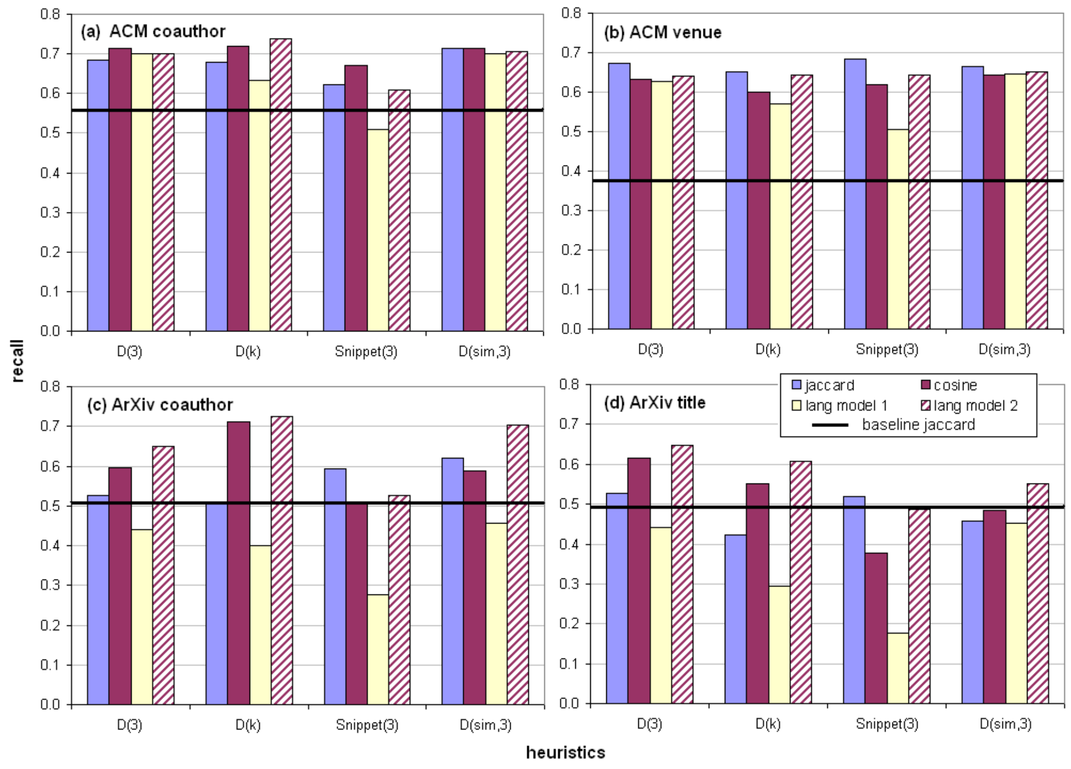


Figure 3.4. Results on the (a) ACM coauthor, (b) ACM venue, (c) ArXiv coauthor, and (d) ArXiv title attributes (all using `Google` as search engine).

venue attributes, respectively. The baseline method performs well on the coauthor attribute but degrades on the other two. The reason for the degradation is that there are usually many tokens in the title attribute of an entity content, and for the venue field there are many common tokens like “conference”, “ieee”, which increase the false positive rate. On the other hand, all of our variations perform better and show around 60-70% recall on all attributes. Our methods show the best on the coauthor attribute too, due to the a more discriminative token selection coming from the last names of the author’s collaborators.

Figures 3.4(c,d) show the results of the ArXiv data set on the coauthor and title attributes, respectively. The baseline jaccard performs around 50% recall on both attributes. Note that most entities to be linked in this set have larger content

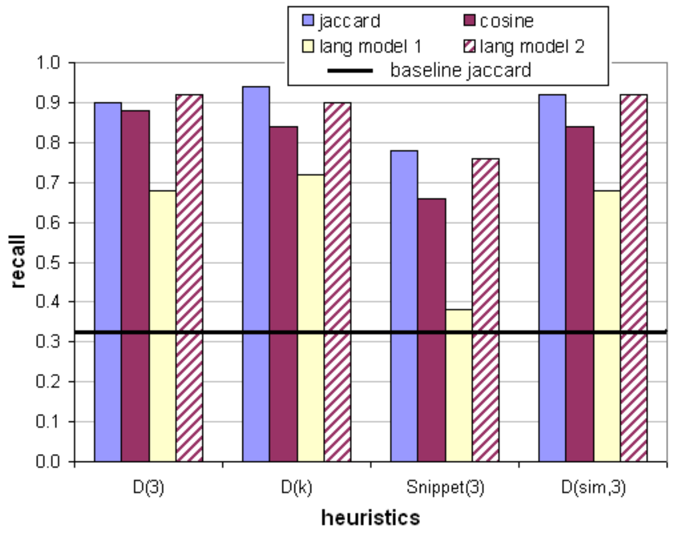


Figure 3.5. Results on the IMDB title attribute using `Google` as search engine.

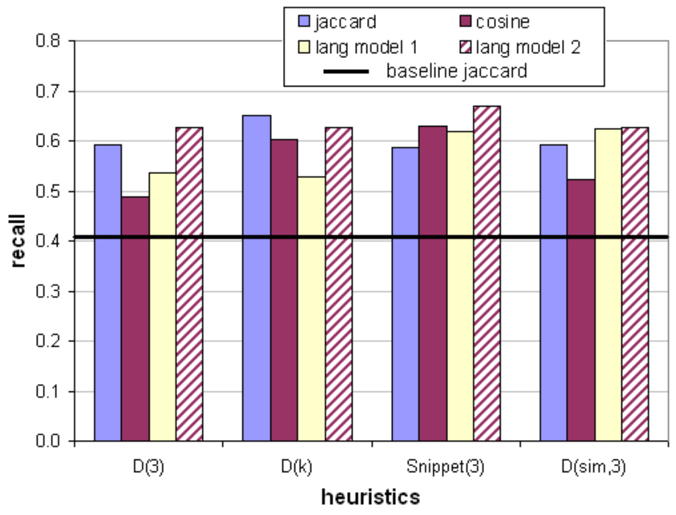


Figure 3.6. Results on the ACM title attribute using `MSN` as search engine.

and lesser number of candidates in each block (Table 4.3), compared to the ACM case. Therefore, it should be easy for any metric to perform well. However, we here attempt to simulate a case where the canonical entities do not have much content and select the canonical entities from each block accordingly, explaining the mediocre recall value of the baseline approach. Although not as good as in the ACM case, our web-based schemes can still have a better recall than the baseline approach, yielding 72% recall at best for coauthor attribute. The smaller improvement on the title attribute stems from undesirable token selection from a small and not-so-good representative title tokens for the entities.

Figure 3.5 show the IMDB result on the movie title attribute. Expectedly, the baseline method shows very low performance having a poor recall value of 32%. Although each actor has enough content (24 titles), such content does not well identify an actor. However, this problem does not apply for the web-based methods which yield 94% recall at best implying that the approach is very promising for such cases.

3. Using Different Search Engine. Our proposed web-based name linkage of using the collective knowledge from the Web is a general approach. As long as it can acquire such knowledge through a search engine, it can have a promising result on the name linkage problem. Thus, we tested how results would change if we use different search engines – `MSN` instead of `Google`. For this set of experiments, we again use the ACM data set and tested the performance on the title attribute using `MSN`. Figure 3.6 shows the new results. The results of the heuristics are consistent with those of Google. While being relatively smaller, all four sets of experiments still outperform the baseline jaccard. The results indicate that as long as we have access to some portion of the Web, we can utilize it to some extend as a collective knowledge source for the name linkage problem.

3.2.4.3 Scalability

Our proposed *web-based name linkage* framework can identify duplicate entities better, especially when entities lack useful information, and complements existing methods well. However, it is not without a problem. Since the proposal relies on

Table 3.4. Running times of the experiments on the ACM data set using the *tf* scheme on the title attribute.

Methods	Recall	Running time	# of Web access
Baseline - jaccard on the current content	0.405	0.09 min	0
Web-based - cosine using *D(3)* and Google	0.653	290.44 min	7951
Web-based - jaccard using *D(3)* and Google	0.711	289.32 min	7951
Web-based - lang. model 1 using *D(3)* and Google	0.573	289.29 min	7951
Web-based - lang. model 2 using *D(3)* and Google	0.756	289.47 min	7951
Web-based - cosine using *D(3)* and the local snapshot	0.463	9.53 min	0
Web-based - jaccard using *D(3)* and the local snapshot	0.443	9.38 min	0
Web-based - lang. model 1 using *D(3)* and the local snapshot	0.457	9.37 min	0
Web-based - lang. model 2 using *D(3)* and the local snapshot	0.494	9.39 min	0

the knowledge acquired from the Web, each reasoning process may incur a large number of web (i.e., network) accesses.

Apart from the computation time, the time spent on accessing the Web information introduces a considerable lag. As an example, using ACM data set and tf scheme, to select a representative token, statistics of response time and the number of required web access are shown in Table 3.4. Note that between subsequent web accesses, we intestinally add 250 msec time-out not to overload search engines. Furthermore, the response time may be affected by many fators – network traffic, load of search engines and web sites, etc.

Due to the huge running time overhead, therefore, the web-based name linkage approach may not be a desirable solution despite the better recall. One solution to alleviate this problem is to use a better *blocking* – i.e., for a canonical entity e_c, instead of examining all entities e_v ($\in E$), pre-select a block of candidates for better performance and accuracy. A more judicious selection of candidates for each canonical entity would be able to reduce the number of candidates significantly, and in turn the number comparisons needed. This would help any linkage solutions but the gain has more obvious impact in the web-based name linkage solutions.

Another solution is to eliminate the need for the Web access by using a local copy of a portion of the Web. The assumption is that as long as the local copy has sufficient coverage of the entire Web, the its behavior may be parallel to that of the entire Web. That is, if an entity e_1 is a duplicate of e_2 since the frequency of e_1 in MSN is similar to that of e_2 in MSN, then the frequency of e_1 in the local copy can be similar to that of e_2 in the local copy. To validate this idea, we use the snapshot from the Stanford WebBase project [41]. The WebBase project provides various crawls of the Web at different times with approximately 100 million pages from more than 50,000 web sites, including many .edu domains. The size of the text crawls is about 1.5TB. From this, we have downloaded the first half of the January 2006 crawl and filtered out irrelevant pages (i.e., pages that do not contain any of the names in our test data sets). Then, we used Apache Lucene[5] to create a local index for each token available in the data. This also provides an interface that mimics the search engine functionalities. At the end, all the access calls to the search engines (e.g., Google and MSN) are re-directed to this "local" search engine with a subset of the Web data – having about 3.5 million pages.

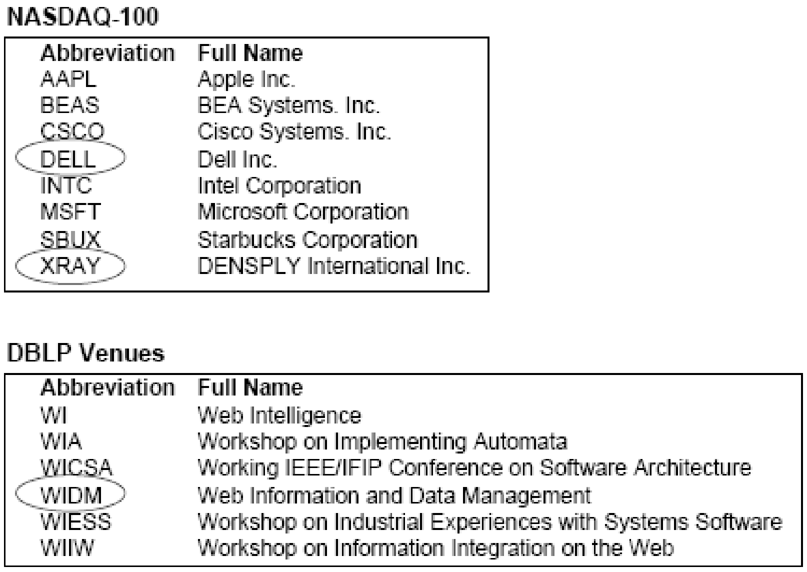
When experiments are repeated using the same data sets, the running times using the local copy decreased dramatically (see Table 3.4). The main reason of the decrease stems from the fact that although the average local search time is almost the same as the case using external search engines, the returned web pages are not needed to be downloaded from the numerous web sites, saving a lot of network costs. Yet, the recall of the new experiments are still reasonably good when compared to the baseline approach, although they are worse by 20-30% than the web-based name linkage solutions using external search engines. Considering the small size of the local snapshot, however, the local search engine result proves

[5]http://lucene.apache.org/

that a sufficient performance can be achieved in a reasonable time once a reasonable size of data sources are used as the collective knowledge source like the Web.

3.2.5 Concluding Remarks

Toward the split name linkage problem to identify duplicate real-world entities with insufficient description or contents, we present a novel approach which exploits the collective knowledge of the Web. Unlike other approaches that use textual similarity of contents or name, our proposal unearths the hidden knowledge from the Web to detect if two entities are the same. Experimental results verify that our proposal improve the recall as high as 193% at best, and outperforms the baseline Jaccard metric for a majority of test cases.



NASDAQ-100

Abbreviation	Full Name
AAPL	Apple Inc.
BEAS	BEA Systems. Inc.
CSCO	Cisco Systems. Inc.
DELL	Dell Inc.
INTC	Intel Corporation
MSFT	Microsoft Corporation
SBUX	Starbucks Corporation
XRAY	DENSPLY International Inc.

DBLP Venues

Abbreviation	Full Name
WI	Web Intelligence
WIA	Workshop on Implementing Automata
WICSA	Working IEEE/IFIP Conference on Software Architecture
WIDM	Web Information and Data Management
WIESS	Workshop on Industrial Experiences with Systems Software
WIIW	Workshop on Information Integration on the Web

Figure 3.7. Various examples of abbreviations [79].

3.3 Linking Short Forms to Long Forms

In this section, we show the effectiveness of our web based linkage approach for the split name linkage problem on a different application, and propose a solution for the scalability problem.

Abbreviations, acronyms, initialisms and shortenings are frequently used in many real world applications due to writing style or for saving space. However, such shortenings also add variations to entity names. When data grows and encompasses multiple resources, this causes critical data quality issues (i.e., split names for entities). Therefore, good solutions for duplicate detection, and record linkage are crucial to maintain the quality of large data collections.

Figure 3.7 illustrates two real world examples of abbreviations for stock ticker symbols in NASDAQ-100 and bibliographic publication venues in DBLP-DL. The figure shows that deducing full names from abbreviations could sometimes be very challenging. For instance, in the case of NASDAQ-100 symbol list, it is relatively easy to guess the full name of the abbreviation “DELL”. However, it is not apparent that “XRAY” is short name for “DENSPLY International Inc”. Moreover, such abbreviations are heavily used in bibliographic citations. In the figure, example abbreviations starting with “W” taken from DBLP-DB are shown. Note that the full workshop name “International Workshop on Web Information and Data Management” is commonly abbreviated to a short abbreviation of “WIDM”.

As illustrated in Figure 3.7, finding linkages between the abbreviations and full names plays an important role in many data applications. Therefore, in this section, we study the problem of linking abbreviations, acronyms and initialisms – hereafter, *short forms* (sf) to (canonical) *long forms* (lf), as an application of our web-based split name linkage solution in the previous section.

3.3.1 Web-based Linkage of Short Forms to Long Forms

Several algorithms exist for detecting abbreviations and their canonical long forms in abstracts and full-text documents (e.g., [6, 18, 75]). However, such algorithms are inappropriate for our problem setting, because they depend on contextual information that is often not available. To remedy this problem, we explore to use our web based linkage framework to get the additional knowledge from the Web. In particular, we use two of our web-based name linkage methods from section 3.2.

Host overlap. This method uses host information of search engine results. We can obtain the hostnames of the top-k search engine results for the short form sf and the long form lf respectively. Then the Jaccard similarity between the sf hostnames and the lf hostnames can be used as a scoring function. We set $k = 10$.

Web page similarity. For each query q, we download the web pages at the top-k

results, and concatenate them into a single text single text document D_q. For a short form sf and a long form lf, the standard *tf-idf* cosine similarity between D_{sf} and D_{lf} is the scoring function. We set $k = 10$.

3.3.2 Evaluation

To validate our method we examine a real-world problem of matching short and long forms in the application area of digital libraries. Our evaluation dataset is a list of short and long form conference and workshop titles from DBLP-DL. It contains 906 short forms and 920 long forms, with 926 short to long form matches. We use Google for our web based linkage methods. To evaluate how accurate and fast the proposed methods are, we use the following metrics:

Average recall and average ranked precision. Suppose a short form sf corresponds to R long forms, and the top-10 candidate long forms in the ranked list for sf contains r correct long forms. Then, the recall is $\frac{r}{R}$. To account for the quality of rankings, we use *ranked precision* [45] instead of traditional precision. Let P_i be the fraction of correct long forms within the top-i candidates. Let C be the positions of the correct long forms within the top-10 candidates. Thus, the ranked precision is $\frac{\sum_{i \in C} P_i}{r}$. The *average recall* and *average ranked precision* are the averages of recall and ranked precision over all short forms in our dataset.

3.3.2.1 Baseline Results

As the problem of venue linkage in our setting has not been studied before to the best of our knowledge, we cannot directly compare our approach against others. Therefore, we establish a baseline by evaluating two state-of-the-art solutions for the similar problem of abbreviation detection that have publicly available implementations: Schwartz and Hearst (S&H) [75] and ALICE [6].

The S&H approach essentially finds a greedy alignment between the short form and the long form, while the ALICE system uses a list of handcrafted rules. These two algorithms are representative heuristic algorithms, but other approaches based on machine learning also exist (e.g., [18, 66]).

In order to use these algorithms in our problem setting, we created a file containing all combinations of short forms and long forms in the dataset, with lines of the form "lf (sf)", which mimics the detection environment for these algorithms. We used this file as input for both implementations. Both algorithms identified a large number of candidate matches (S&H and ALICE identified 43771 and 74596 matches respectively, but our dataset has only 926 true matches). As both algorithms did not output any matching quality score, but instead indicated which part of the long form was matched (e.g., the "Key Cryptography" part of "Public Key Cryptography"), we set the scoring function to be the fraction of characters in the long form that was matched. For the above example, this fraction is $\frac{15}{21} = 0.714$.

3.3.2.2 Results

Method	R	P
S&H	0.532	0.244
ALICE	0.393	0.148
Host overlap	0.641	0.545
Web page similarity	**0.802**	**0.714**

Table 3.5. Results for the baselines and web-based methods.

The results for using the baselines and the web-based methods are shown in Table 3.5. We found that S&H has average recall of 0.532 and average ranked precision of 0.244, while ALICE has average recall 0.393 and average ranked precision of 0.148. The rather poor performance indicates that neither algorithm is suitable for our problem setting and they significantly suffer from lacking any additional context for linkage. Our web-based methods, on the other hand, significantly outperform the baseline methods meaning that both short forms and long forms are used interchangeably on the web thus providing plenty of relevant information for the linkage process.

3.3.3 Adaptive Querying

Our proposed framework can link short to long forms very effectively with a variety of methods. However, some methods require expensive downloading and processing of web pages. Our experimental results (Figure 3.8) for the top-10 search engine results shows that web page similarity obtains about 25% and 31% better recall and ranked precision than host overlap. However, web page similarity scheme takes up to three orders of magnitudes longer to run, significantly reducing its practicality.

Algorithm 1: Adaptive querying.

```
Require: a stronger method $M_s$, a weaker method $M_w$ and a heuristic $H$
  for all $sf \in SF$ do
    obtain top-$k$ $lf$ results using $M_w$;
    if $H$ determines that $M_w$ results are likely incorrect then
      obtain top-$k$ $lf$ results using $M_s$;
      report $M_s$ results;
    else
      report $M_w$ results;
```

Adaptive methods combine base methods to obtain the better aspect of each, and is adopted in data integration work (e.g., [62, 88]). [46] proposed an adaptive framework to determine when to search or when to crawl. Algorithm 1 shows the general scheme of our adaptive querying algorithm: for each problem instance, we first obtain initial results using a faster but less accurate method M_w, and use a heuristic H to determine whether we need to re-run the problem using the slower but more accurate M_s. Thus, we save overall execution time by making fewer calls to M_s. In our setting, we can combine the high accuracy of web page similarity (our M_s) with the quick execution time of host overlap (our M_w). We observe that if host overlap returns the correct long form result for a short form sf, it is usually returned as the first result, and the remaining results usually have considerably lower similarity scores. As the other long forms usually have only one or two common URLs with sf, the variance in their scores is very small. Let r_1 and r_2 be the top two results for a sf. We thus define five heuristics:

Heuristic 1. Let sim_1 and sim_2 be the similarity scores of r_1 and r_2 respectively. If $(sim_1 - sim_2)/sim_2$ is below a threshold, i.e., sim_1 and sim_2 is not significantly different, then we use web page similarity instead.

Heuristic 2. If the number of common URLs between the r_1 and sf results is below a threshold, say two or three, then web page similarity is used.

Heuristic 3. If the average IHF [78] weight of the common URLs between sf and r_1 is less than some threshold, web page similarity is used.

Heuristic 4. Let h be the common URL between r_1 and sf with the highest IHF weight. If IHF(h) is below a threshold, then it is not discriminative enough, and we turn to web page similarity for this case.

Heuristic 5. If the average IHF weight of sf's URLs is less than a threshold, we use web page similarity. The URLs of the long forms are not used here.

In summary, Heuristics 1 and 2 attempt to characterize possible matches by examining the returned scores, while Heuristics 3 to 5 rely on information on the URLs and also incorporate collection statistics, e.g., IHF.

3.3.4 Adaptive Querying Results

We evaluated our adaptive querying technique using web page similarity (stronger method M_s) and host overlap (weaker method M_w), both using top-10 results. For each heuristic, we experimentally determine its threshold by analyzing a sample set of results. Figure 3.8 shows the performance of these heuristics.

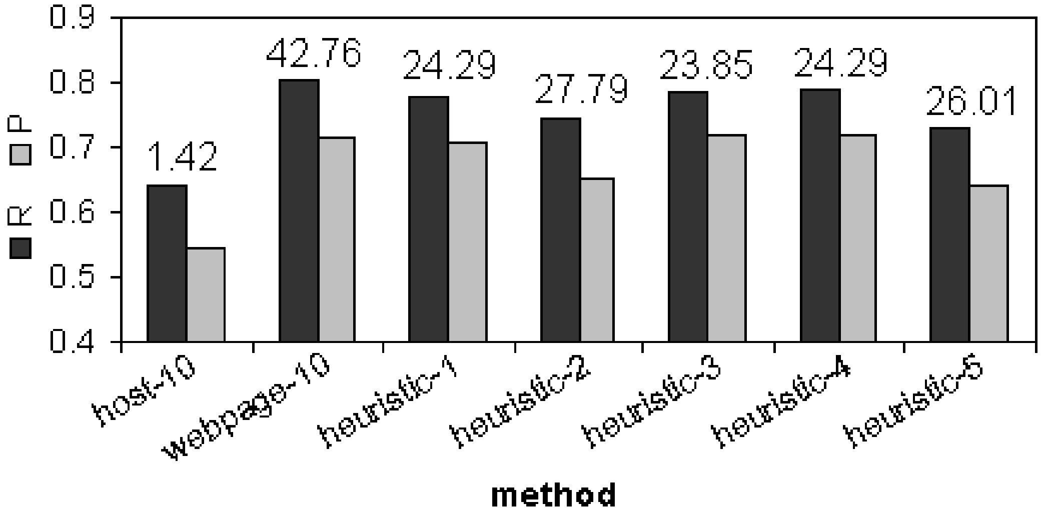


Figure 3.8. Performance and running time results for host overlap, web page similarity and the adaptive methods, all using top-10 results. Value above bars indicates the average time required to execute each short form case in seconds.

The best heuristic is Heuristic 4. Its recall (0.788) is close to web page similarity, and its precision (0.719) beats web page similarity. As it executes web page similarity for only 40% of the short forms, it is 43.8% faster than web page similarity, with only a small 1.73% drop in accuracy.

Heuristics 3 and 4 are similar and achieved similar performance, and show that a common rare URL in the short and long form results gives strong linkage evidence. Recall of Heuristic 5 suffers because only short forms are considered. Un-weighted URL counts (Heuristic 2) is less effective as many "useless" common URLs, such as aggregator web sites, may be shared. Heuristic 1 is almost as effective as Heuristics 3 and 4, but did not have better recall because it could not correctly identify some challenging correct matches.

These experiments shows that adaptive querying can create a more scalable algorithm whose performance is similar to the stronger method.

3.3.5 Concluding Remarks

Short forms are frequently used in many real world applications in place of long forms which makes linkage of records very changeling. Approximate string matching do not tend to work well due to these differences, and motivate our approach of using the Web as the additional evidence. In this study, we showed the effectiveness of our web-based methods over the baseline approaches. Moreover, we presented a way optimizing the amount of time and network bandwidth needed to solve this linkage problem: an adaptive querying algorithm that combines a stronger algorithm that runs slowly and a weaker algorithm that runs much faster, so that the performance is near the stronger algorithm yet takes much lesser time. This technique is a very general framework and can result in significant savings in running time and network traffic, and are complementary to each other.

Chapter 4

Improving Precision in the Split Name Linkage

In this chapter, we attempt to tackle the split name linkage problem by discovering semantic knowledge hidden in contextual information of entities. Although traditional linkage approaches work well in many scenarios (reviewed in Chapter 2), they often suffer from a large number of *false positives* (an entity determined to be a variant when it is not). If a name linkage system returns top-k answers, such false positives may even override correct variants out of the answer window of $|k|$. This causes system recall and especially precision to degrade substantially.

Unlike regular entities, a named entity often contains *a group of tuples* in it, such as an author entity with a paper list. For instance, two address entities, e_1 and e_2, may be detected as variants if their corresponding properties, ("East College Ave.", 16801, "PA") and ("E. College Ave.", 16801, "Pennsylvania"), are similar. For this kind of the split name linkage problem, we can capture the group of tuples of two entities into some data structure, and measure the distance between between the two data structures. Typically, the group of tuples of a named entity is structured and follows a certain format. However, such tuples are likely composed of a large number of tokens not all of which are directly a good indicative of the entity. This may confuse regular distance functions and result in false positives [65].

The false positive problem stems from the fact that distance functions used in linkage systems 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 can improve precision significantly. Our method unearths the relationship hidden under named entities, and exploits this semantic knowledge together with textual similarities.

This chapter is organized as follows. We first introduce the formal definition of the problem. Then we show how we create the collaboration network of computer scientist to mine the required semantic knowledge for our method and analyze the collaboration network for many characteristics that can be used for the linkage process. The later section describes our solution in detail based on the notion of finding quasi-cliques on graph representation of entity context. Finally, we propose a network model for the collaboration network we created to improve the quality of the collaboration network for the linkage process, i.e., anticipating missing links etc.

4.1 Mining for Hidden Knowledge from Entity Context

Entity context is often more than a simple textual information related to the entity in question. It may be composed of more valuable information, and possibly structured, made of entities with different types. By understanding the relations and patterns between these entities inside the context, more descriptive information can be extracted to be used in the linkage process. In this section, we study various ways of extracting the hidden information from entity context to target the false positive problem in the split name linkage.

Let us consider a split name linkage problem frequently occurring in digital libraries. As described in chapter 3, the scenario includes authors as entities and the citations of the authors as entity context. Consider the coauthor information of the citations. Each author has a set of coauthors that he or she has collaborated. Regarding this simple list of coauthor information more than some textual information, we can also find out and extract the collaboration relations between all the coauthors and the author in question. Further, we can use all these new information that we extracted along with the textual information for our decision in the linkage process. Now, we describe how such relations and patterns are mined from entity context and what different types of such hidden information could become available for the linkage process.

Social network analysis is an active research field in the social sciences where researchers try to understand social influence and groupings of a set of people or groups. Its origin is in general believed to be due to Milgram [59] who identified the so-called "*six degrees of separation*" phenomenon based on the results of an experiment: any two people in the United States are connected through about 6 intermediate acquaintances, implying we live in a rather *small-world*. Since then, sociologists and psychologists have found evidence for a wide range of small-world phenomena arising in other social and physical networks (e.g., power grids, airline time tables, food chain, World-Wide Web, Erdös number). Inspired by some of the recent attempts to apply social network analysis to scientific communities [61, 10, 60, 27], in this section, we analyze the collaboration network made of the researchers in the computing area at large. We search for interesting patterns underlying the computing community and their publication behavior that might give insight and helpful clues for utilizing the hidden information in the collaboration network for the split name linkage problem.

The Guide (http://portal.acm.org/guide.cfm) is a collection of bibliographic citations and abstracts of works published by *Association for Computing Machinery (ACM)* and other publishers. It currently has more than 750,000 citations from 3,000+ publishers, covering books, journal articles, conference proceedings, dissertations, theses and technical reports. It is automatically generated by extracting metadata from those resources. Since it is a high-quality citation digital library that has a very good coverage on computing literature, we chose to use The Guide as the data set for our analysis of the computing community. In particular, we examined citation data in The Guide from 1950 to 2004 (obtained by crawling The Guide in January 2005), which contained about 609,000 authors and 770,000 publications.

We use a *collaboration network* (or graph) where nodes represent authors and edges between any two nodes exist if those nodes represent authors who have co-authored one or more papers (about 1.2 million edges). Note that The Guide itself does not have a notion of "unique key" such as DOI (Digital Object Identifier). Instead, it depends on the name of authors to distinguish them. Therefore, the classical *name authority control problem* [83] may arise (i.e., same author with various spellings or different authors with the same spelling). We try to minimize the effect of this problem by conducting two experiments - one with full names ("John Doe") and the other with the first initial followed by the last name ("J. Doe") - and use these as the upper and lower bounds of the statistics. For the visualization of our network analysis, we used Pajek [12], a freely available social network analysis tool for non-commercial use.

4.1.1 Basic Statistics

First, we present various statistical analysis related to the authors. Figure 4.1a shows the number of "new authors" (ones who publish a paper for the first time in the venues covered) as a function of year. There is a small community in the first 30 years, this is due to the coverage of the database and less number of venues available then. In the late 1980s, the increase rate becomes around 40% and the community steadily grows. In the recent years, however, the same pace cannot be maintained, about 7% increase each year with slowing tendency. There are roughly 28,000 new authors in 2004 alone who publish a paper for the first time – novice graduate students or veteran scholars from related fields. Although the community is very large, each year, only a small portion (about 10%) of all authors are "active authors", i.e., ones who publish at least one paper in that year. Interestingly, almost half of the active authors in any given year are "new authors" and they are steadily contributing to about 55% of papers each year in the recent years. The remaining part is contributed purely by the existing authors, which increases

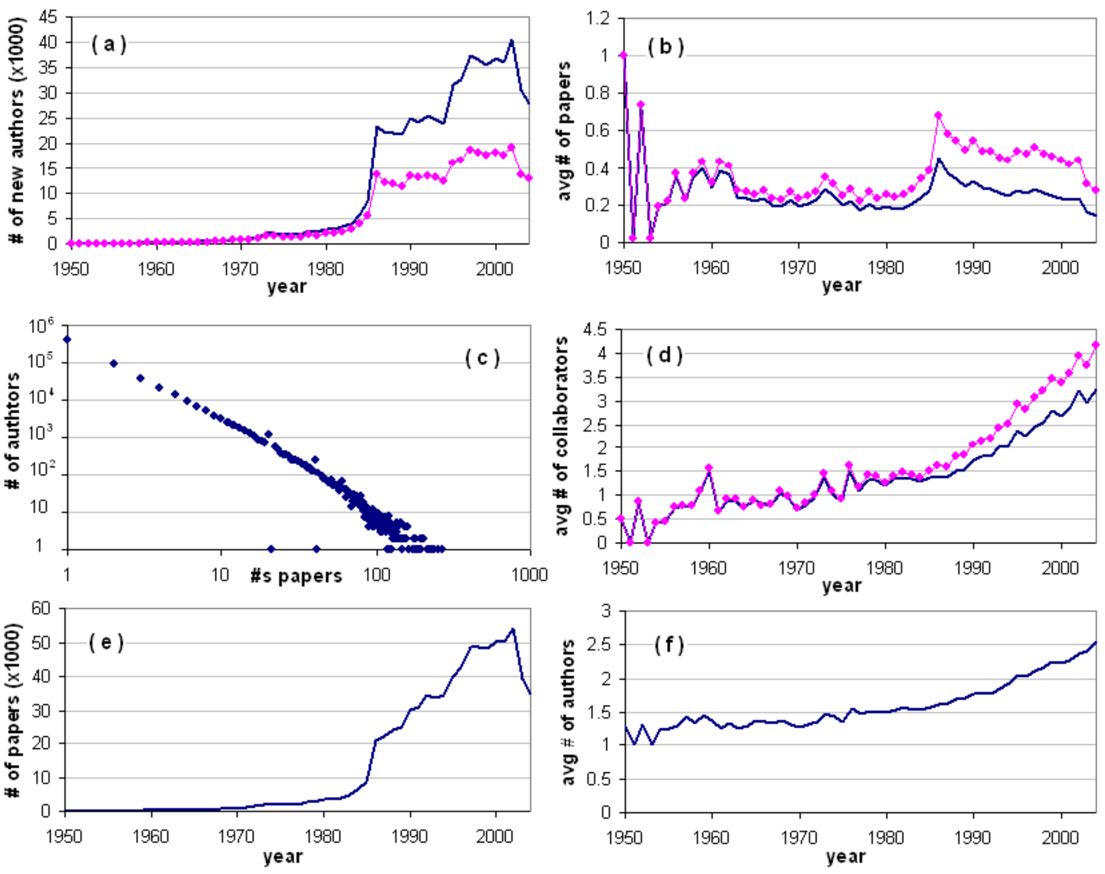


Figure 4.1. Basic Statistics (fig a-d: blue-solid lines for full name set, pink-dotted lines for abbreviated name set): (a) # of new authors per year, (b) avg. # of papers per author, (c) distribution of # of papers per author, (d) avg. # of collaborators per author, (e) # of papers per year, (f) avg.# of authors per paper.

the density of the network by creating new edges through new collaborations. Figure 4.1b illustrates the average number of papers per author for a given year. It does not seem to change dramatically, implying that the productivity rate of the computing community as a whole remains intact over time. This makes sense since only small fractions of the community are active each year and they can publish only a limited number of papers.

Lotka's Law describes the frequency of publications by authors by "*the number of authors making* n *contributions is about* $1/n^2$ *of those making one; and*

number of papers	**number of co-authors**	**closeness**	**betweenness**
266 B. Shneiderman	229 J. Dongarra	0.18885 S. Muthukrishnan	0.008837 B. Shneiderman
250 M. Karpinski	204 A. Gupta	0.18785 G. Wiederhold	0.008506 E. Bertino
236 H. Garcia-Molina	193 B. Shneiderman	0.18778 H. V. Jagadish	0.007474 L. T. Watson
234 P. S. Yu	181 I. Foster	0.18754 C. Faloutsos	0.007218 G. Wiederhold
226 M. Sharir	180 H. Garcia-Molina	0.18701 H. Garcia-Molina	0.007194 J. Dongarra
225 M. Stonebraker	175 E. Bertino	0.18629 E. Bertino	0.006921 M. Fujita
217 K. G. Shin	175 M. Stonebraker	0.18584 V.S. Subrahmanian	0.006799 S. Muthukrishnan
204 E. Bertino	174 G. Wiederhold	0.18572 M. Stonebraker	0.006697 W. Li
204 G. B. Shelly	174 R. Kikinis	0.18562 J. D. Ullman	0.006578 N. Alon
196 P. J. Denning	169 N. Alon	0.18559 U. Dayal	0.006246 C. Faloutsos

Table 4.1. The top-10 authors with the highest number of papers, number of co-authors, closeness and betweenness scores.

the proportion of all contributors, that make a single contribution, is about 60 percent" [52]. He showed that such a distribution follows a power law with an exponent approximately -2. Figure 4.1c shows the distribution of numbers of papers per author on log-log scales for our database, of which the exponent is -2.59. Consistent with Lotka's Law, a small number of authors publish a large number of papers whereas 392,559 authors (64%) have only one paper (the fat tail on the right hand side indicates this). In fact, in ACM database, there are only 36 authors who published more than 150 papers. Top-10 authors with the most number of publications are shown in the first column of Table 4.1.

Now, we examine the number of collaborators an author has, illustrated in Figure 4.1d. The average number of collaborators per author tends to increase steadily, 3.66 for the cumulative data up to 2004. Compared to other scientific communities having large-scale experimentation with large groups (e.g., high-energy physics), this average is rather small. The steady increase of the average number of collaborators can be hypothesized as follows: (1) the so-called "Publish or Perish" pressure drives scholars to seek more effective ways to increase the number of publications such as collaborative research; and (2) the rapid development and deployment of new communication mediums such as email, instant messaging, video conferences make remote collaborations easier than before. For instance, the last author of this

Similar Venue Pair		Distance
Journal of Algorithms	Symp. on Discrete Algorithms	0.19951
SIAM Journal on Computing	Ann. ACM Symp. on Theory of Comp.	0.18474
Comp. Linguistics	Ann. Meeting of the ACL	0.18042
Comp. Geometry: Theory & Appl.	Ann. Symp. on Comp. Geometry	0.18039
SIAM Journal on Computing	Symp. on Discrete Algorithms	0.17260
Ann. ACM Symp. on Theory of Comp.	Symp. on Discrete Algo.	0.16362
J. of Combinatorial Theory Series B	J. of Graph Theory	0.16018
J. of Computer and System Sciences	Ann. ACM Symp. on Theory of Comp.	0.15634
ACM TOPLAS	Ann. Symp. on Principles of Prog. Lang.	0.15551
J. of Functional Prog.	Int. Conf. on Functional Prog.	0.15447

Table 4.2. Top 10 pairs of venues with the highest Jaccard distance.

article has collaborated with the others via emails. The distribution of collaborators exhibits the power-law tail as well with exponent -2.84. The second column of Table 4.1 shows the authors with the largest number of collaborators.

The number of papers published each year (Figure 4.1e) shows very close relationship with the number of active (and new) authors verifying that the productivity is constant on average. In 2004 alone, there are more than 34,000 papers published, largely due to the increased number of authors. The average number of authors per paper tends to increase each year, yielding almost 2.6 co-authors per paper as of 2004 (Figure 4.1f). Although there are a significant number of papers with only a single author, the majority of papers are written by two authors (13557 papers). The figure clearly shows that there is an increasing tendency for collaboration among authors which also causes papers to have more co-authors.

Next, we looked at how publication venues are inter-related to each other using co-authorship information. By examining the pattern where the scholars publish their papers, one can see, for instance, which publications venues have a similar theme or orientation. Figure 4.2a shows a graph where (1) a node is a publication venue, whose size is proportional to the number of papers in it, and (2) an edge between venues X and Y reflects the similarity by the Jaccard distance, $\frac{|A \cap B|}{|A \cup B|}$, where A and B are author sets of venues X and Y. Hierarchical organization of

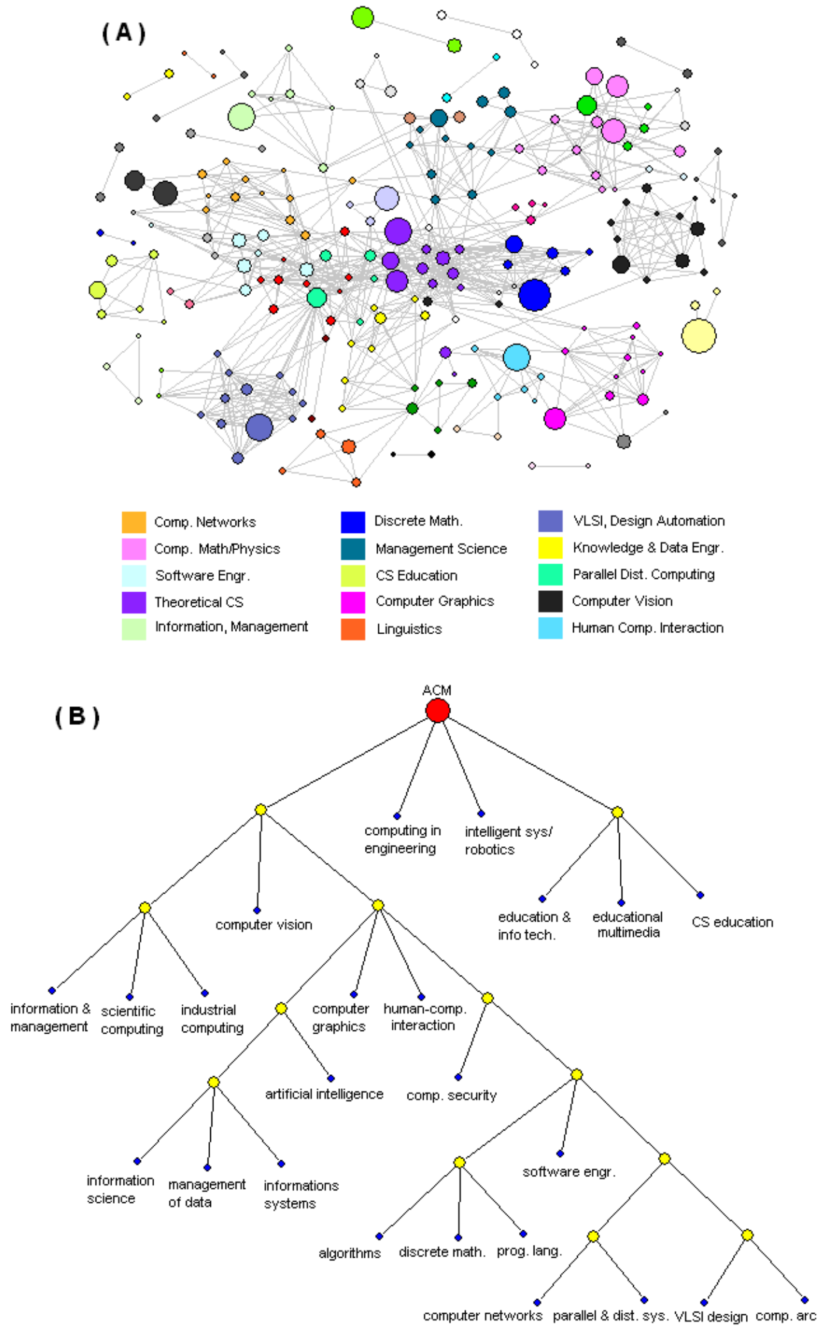


Figure 4.2. (a) Venue relations (only venues with at least 300 papers and edges with at least 0.3 Jaccard distance are shown). The size of a node is proportional to the number of papers in the venue [64]. (b) Hierarchical organization of the fields that have strong relations with each other according to Jaccard distance.

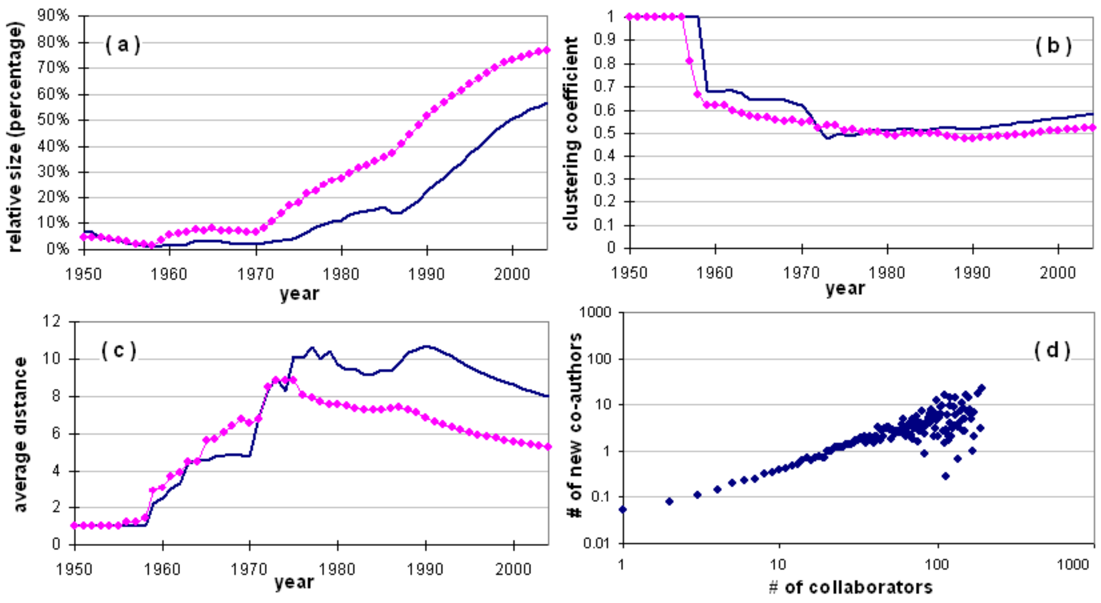


Figure 4.3. Structural Properties (blue-solid lines for full name set, pink-dotted lines for abbreviated name set): (a) relative size of the giant component, (b) clustering coefficients, (c) geodesics, (d) preferential attachment.

the fields that have large venues with highly overlapping authors between each other is shown in Figure 4.2b while Table 4.2 lists the 10 pairs of the computing publication outlets having the highest similarity measure.

4.1.2 The Giant Component

The giant component of a graph is the largest subset of interconnected nodes in the graph. The rest of the nodes usually form much smaller components, typically of size $O(\log n)$, where n is the total number of nodes [61]. Figure 4.3a shows the relative size of the giant component in our collaboration graph, which is the ratio of the nodes in the component to the all nodes in the graph. In the initial years, the size of the giant component of the graph is much smaller compared to the total number of authors, covering only about 3% of the whole graph although new authors keep joining to the community. Yet, those authors help cluster other large components in the graph. In the mid 1970s, those clusters start to form larger components. The giant component then steadily grows till 2004. This is because new authors keep joining and existing authors collaborate more, smaller components are merged to the giant one.

As of 2004, the size of the giant component is 346137, 57% of the entire community. However, the second largest component is much smaller; it includes only 66 authors, who work on very particular subjects and publish mostly in "Cybernetics and Systems Analysis" journal. The collaboration graph also has 727 "*minor*" components with 10-19 authors and 5,300 components with 5-9 authors. The reason for that many small components with remarkable number of authors might be due to: (1) geographical isolation of the research institutions since the database covers venues world wide, (2) variety of the subjects which have no overlap with the others, and (3) possibly name ambiguity problem [63].

4.1.3 Clustering Coefficients

Given a node v, the *neighborhood* of v, $N(v)$, is a subgraph that consists of the nodes adjacent to the node v. Furthermore, let us denote the edges and nodes in $N(v)$ by $E(N(v))$ and $K(N(v))$, respectively. Then, the clustering coefficient of v, $\gamma(v)$, is:

$$\gamma(v) = \frac{|E(N(v)|}{|E_{max}(N(v))|}$$

where $|E_{max}(N(v))| = \frac{|K(N(v)|(|K(N(v))|-1)}{2}$ (when the neighborhood is fully-connected – clique) [85]. Therefore, the clustering coefficient measures how many edges actually occur compared to the fully-connected case. The clustering coefficient of a graph G, $\gamma(G)$, is the average clustering coefficients of all nodes in G. This measure could be viewed as the degree to which a scholar's collaborators have collaborated with each other.

The evolution of the clustering coefficient of the giant component is shown in Figure 4.3b. First few years result in a fully connected graph; since the size of the component is small everyone has acquaintance with each other. Over the following years, the clustering coefficient tends to decrease as the giant component expands. Starting from the year 1974, when the giant component starts becoming relatively larger, the clustering coefficient shows a steady increase thereafter yielding a final value of 0.6. In other words, on average 60% of one's collaborators are connected to each other. This high value implies that the network should have a hierarchical structure. The network should mostly be composed of scientific groups or communities of different sub-fields that are highly interconnected within the group but have only a few links to nodes outside of the group to which they belong to. Hence, there are usually many nodes that have dense connections only within their community, where as relatively small number of other nodes creates the hierarchical topology by having connections between communities.

4.1.4 Geodesics

In a co-authorship network, two authors could know each other through their collaborators. The path(s) with the minimum number of edges between any given pair of authors in the network is called the shortest path or geodesic of the pair. Then the average distance in a network is the average of all pair-wise geodesics in the network. Social networks often have small average distances compared to the number of nodes in the networks, first described by [59] and now referred to as "small-world effect". Figure 4.3c shows the evolution of the average distances in the giant component of the community.

In the first forty years of the period analyzed, the geodesic tends to increase each year with occasional fluctuations. After it finally reaches its maximum, 10.7, in 1990, it decreases continuously. Although the community becomes larger and larger in size by addition of new authors, this decrease in the geodesic implies that new more collaborations tend to be occurring between existing authors which create new shorter paths between all authors through the network, prevailing the effect of expansion due to the new authors.

The final value of the geodesic in 2004 is 7.9 and seems to be decreasing in the following years ("8 degrees of separation"). Compared to the size of the community, this low value is probably a good sign since scientific discoveries can be disseminated rather fast [61]. The diameter of a graph, the maximum of the pair-wise distances in the giant component, of the computing community is 33 as of 2004.

4.1.5 Centrality

An interesting analysis of co-authorship network is to identify the most "central" scholars of the community. Authors who are the most prominent in the community are often (certainly not always) located in the strategic locations of the co-authorship network. Such a location may allow one: (1) to communicate directly with many other authors, (2) to be close to many other authors, or (3) to be as intermediary in the interactions of many other pair of authors.

Closeness centrality is one of the common methods used to to quantify authors' locations [84]. The *closeness* can be defined as how close an author is to all other authors on average. Authors with low values of this average could be viewed as those who can access new information quicker than others and similarly, information originating from those authors can be disseminated to others more quickly [61]. Formally, the closeness of a node v in a connected graph G is defined as follows:

$$C(v) = \frac{n-1}{\sum_{n,v \in G} d(v,w)}$$

where $d(v,w)$ is the pair-wise geodesic and n is the number of all nodes reachable from v in G. That means that, it is 1 over the average of the shortest paths from v to all other nodes in G. The third column of Table 4.1 lists the top-10 individuals according to the closeness scores.

Sometimes the interactions between any two non-directly connected authors (i.e., who never collaborated before) might depend on the authors who connect them through their shortest path(s). These authors potentially play an important role in the network by controlling the flow of interactions. Hence the authors who lie between most of the shortest paths of the pairs of authors could be viewed as the central people in the community. This notion, known as the *betweenness centrality* of a node v, $B(v)$, measures the number of geodesics between pairs of nodes passing through v, and formally defined as follows [32]:

$$B(v) = \sum_{v,w,x \in G} \frac{d(w,x;v)}{d(w,x)}$$

where $d(w,x)$ is a geodesic between w and x, and $d(w,x;v)$ is a geodesic between w and x passing through v. The equation can be also interpreted as the sum of all probabilities a shortest path between each pair of nodes w and x passes through node v. The fourth column of Table 4.1 shows the top-10 authors with the highest betweenness scores. The top authors in both rankings are indeed prominent scholars in the computing community.

4.1.6 Preferential Attachment

Preferential attachment, proposed by [9], is an interesting phenomenon for modeling network growth. The basic assumption underling this phenomenon is that the probability that a node in a network gets new links is proportional to the number of links it already has, leading to a multiplicative process, which is known to give the power-law connectivity distribution. Due to its simplicity, several extensions have been widely used to model the growth of complex networks using only link structure [10], or both link and node properties [58].

For our social network, it implies that an author with many co-authors acquires new collaborators with a higher rate than one with less number of co-authors. In order to test if such a phenomenon exists, one needs temporal data, where the exact order that each particular collaboration is known. Then by making a histogram of the nodes with k links to which each new link is added, we have a function of preferential attachment. This should be an increasing function of k, believed to be a linear function in its basic form so that the resulting connectivity distribution (distribution of number of collaborators) will exhibit power-law [9]. Figure 4.3d shows the preferential attachment in our network. Since we have only data dated to the year, we present the change Δk in the number collaborators within one year interval (2000 - 2001), for an old author who had k collaborators at the beginning of the previous year (2000). Although the precision is not ideal, it exhibits preferential attachment behavior. The “rich” (ones with many co-authors) get “richer” (acquire more new co-authors than the others). In the case of no preferential attachment, the function would be constant, i.e., each author acquires new collaborators with the same probability.

4.1.7 Concluding Remarks

Entity context such as coauthor information of an author's citation list includes much more valuable clues than a simple textual content for the author. By unearthing such information from entity context, we can perform more accurate linkage and alleviate the false positive problem. In this section, we search for different patterns and measures in the collaboration network of scientists who publish in the computing literature towards this goal. We present a list of statistics and structural properties for the collaboration network that might be incorporated to a linkage framework based on textual similarity for a more effective name linkage process.

4.2 Graph-based Split Name Linkage

Towards the solution for the false positive problem in name linkage, in this section, we propose a graph based method to capture contextual information and find relationships as additional knowledge for more effective linkage. Our approach relies on the following hypothesis:

"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" [48].

For instance, in the 'Ullman' example of Figure 1.1(a), the canonical name "J. D. Ullman" has a name variant "J. Ullman". In order to avoid linking another author "J. K. Ullman" to "J. D. Ullman", we may discover the hidden relationships between two and utilize this information as additional knowledge. Note that relying purely on textual similarities on the co-author lists of two authors may likely suffer from the false positive problem. Although both are database researchers, their frequent co-author lists are slightly different. If such relationships, i.e. co-authorship, could be represented as graphs, then the distance between two graphs can be used to measure the distance between the two entities. Since graphs, i.e., collaboration networks, may contain richer information, such as relationships between nodes, than simple textual representation as studied in the previous section, the distance measurement from graphs are likely to help more in identifying real name variants and therefore alleviating the false positive problem.

4.2.1 Capturing Context as a Graph

We consider the name linkage case where each entity has a group of elements associated with it. For instance, an author name may have a set of his co-author names, a token of word used in his paper titles or a set of venue names in which

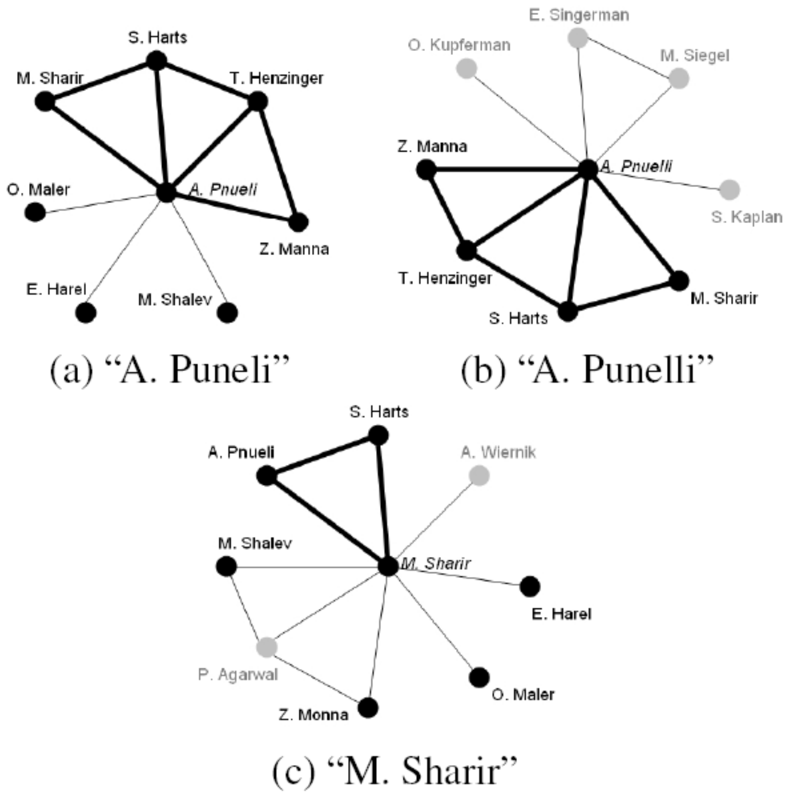


Figure 4.4. Graph representations of the "contexts" (i.e., co-authors) of three named entities, and their superimposed quasi-cliques (solid lines) [65].

his papers published etc. We call all these additional information associated with an entity name as **context** of the entity. When this contextual information is captured as a graph, let us call it as **context graph**.

Consider a real-world example of a split name linkage case drawn from the ACM digital library. There are three entities, E_a ("A. Puneli" entity), E_b ("A. Punelli" entity), and E_c ("M. Sharir" entity), of which E_a and E_b are name variants and E_c is a false positive of E_a [65]. The co-author list of each authors in the above example is as follows: In ACM data set, each entity has the following co-authors lists [65]:

- 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"}

Suppose that we compare these entities only by textual similarity in their context (i.e., the list of co-authors). Let us have a textual distance function that measures how many common co-authors two entities have. Then, this distance measure returns 5 and 7 common co-authors with E_a for E_b and E_c, respectively. Therefore, E_c would be returned as the name variant of E_a creating a false positive.

However, we can make a better decision for the linkage of these entities if we use more information from their context. If we represent each entity context as a graph where the entity itself becomes the center node and its co-authors become neighboring vertices attached to the center node, then we get Figure 4.4 [65]. Furthermore, if there are known collaborations 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 4.4(a) [65].

Suppose that we now compare three entities in terms of how large the maximum common subgraph (where we attempt to match 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 [65]. Then such a distance function would return E_b, the correct name variant of E_a, hence eliminating the false positive problem of the previous case.

As this example illustrates, we propose to use a graph-based partition to tackle the split name linkage problem to alleviate the false positive problem of the general textual based similarity techniques.

4.2.2 *Quasi-Clique*

Once textual context of two entities are captured into two graphs, the problem turns out to how to measure the distance between the two graphs. In order to scale the solution, we propose to use a graph mining technique using the notion of *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 $E_U = \{(u, v)|(u, v) \in E(G) \wedge u, v \in U\}$ [68].

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) [68].

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 [68].

A *Quasi-Clique* in a graph may strongly indicate a group of objects highly interactive with each other. Thus it might be more reliable to use such objects included in the *Quasi-Clique* to represent the corresponding entity than the other objects in the graph.

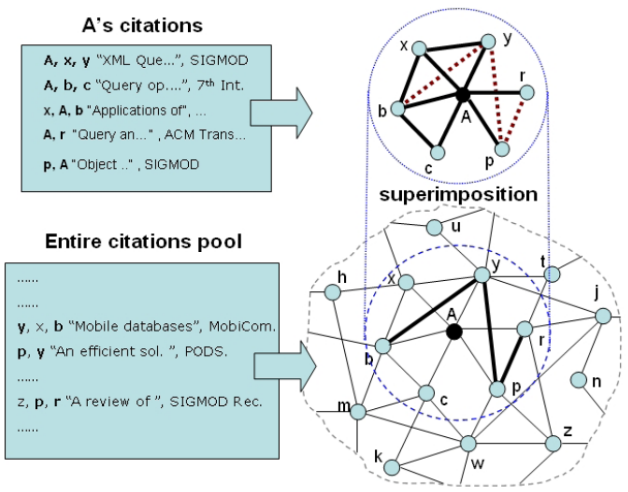

A's citations
A, x, y "XML Que...", SIGMOD
A, b, c "Query op....", 7th Int.
x, A, b "Applications of", ...
A, r "Query an..." , ACM Trans...
p, A "Object .." , SIGMOD
Entire citations pool
y, x, b "Mobile databases", MobiCom.
p, y "An efficient sol. ", PODS.
z, p, r "A review of ", SIGMOD Rec.
superimposition

Figure 4.5. Capturing the entity context into a graph [65].

4.2.3 The Graph Based Name Linkage Algorithm

We now introduce our graph based algorithm to tackle the name linkage problem from the perspective of eliminating the false positive problem. Our algorithm consists of two steps: (1) we first capture the contextual information into a graph, (2) we then measure the similarity between context graphs for linkage decision.

In step 1, our goal is to represent the context of an entity as a graph. Consider an author entity A and its co-author set $\{ b, c, c, p, r, x, y \}$ as its textual context. Then, using a collaboration network we can capture this information into a graph. As seen in the upper portion of Figure 4.5, such a graph represent the canonical entity A as a middle vertex and each of its co-authors as another vertex in the graph connected to it, representing the collaboration between A and its co-author. Other collaborations between co-authors in A's citations are also captured here. Suppose that A, b, and c have collaborated on a paper, then there is an edge available between b and c in the graph, as seen in the figure.

Note that we can create the graph by using A context alone, i.e., its co-authors from A's citations. However, we can even make the graph have richer information.

Consider a global collaboration network created by using the entire citations available in the data set in question (the lower portion of Figure 4.5). Then, unseen collaborations between A's co-authors in A's citations can be discovered from this global collaboration network, i.e. a base graph, and missing connection could be completed between A's co-authors in A's context graph. the process what we call as "superimposition" (Figure 4.5).

We are not limited with the co-author attribute to use our proposal. Suppose the venue attribute of a citation data set. We can create a base graph of all venues in the data set in which vertices are venues and edges represent some relation between the venues, i.e., number of authors who published in both venues. For more general case, a base graph can be constructed using the distinct tokens in the attribute in question. In such a case, the relation between tokens could be the co-occupance pattern of the tokens on the same attribute value, i.e., the probability that two tokens appear in the same publication title etc.

Algorithm 2: `distQC`

```
Input  : A grouped-entity $e$, an ER method $M$, and three parameters ($\alpha$,
         $\gamma$, and $S$).
Output: $k$ variant grouped-entites, $e_v(\in E)$, such that $e_v e$.
1 Using $M$, find top $\alpha \times k$ candidate entities, $e_X$;
2 $G_c(e) \leftarrow$ context graph of $e$;
3 forall $e_i(\in e_X)$ do
4 $G_c(e_i) \leftarrow$ context graph of $e_i$;
5 $g_i \leftarrow QC(G_c(e), \gamma, S)$;
6 Sort $e_i(\in E_X)$ by $|g_i|$, and return top-$k$;
```

In step 2, we propose a distance metric distQC to assess the similarity between two entity contexts using the notion of quasi-clique. The algorithm is shown in

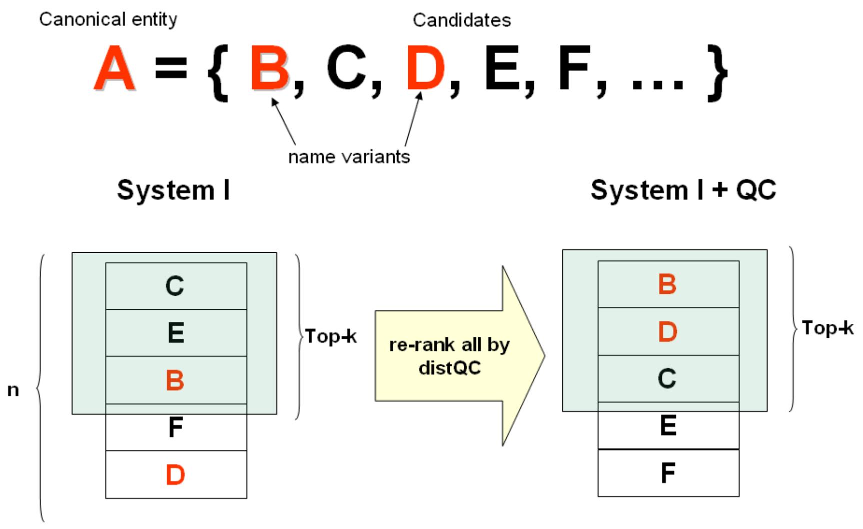


Figure 4.6. Illustration of our system is combined with another linkage framework.

Algorithm 2 [65]. There are two important factors to consider in such a distance function. First, the γ parameter determines indicates the compactness of a γ-*Quasi-Clique*, two entities in a *Quasi-Clique* with a higher γ value are more similar to each other than two in a *Quasi-Clique* with a lower γ value. Second, the size of the common *Quasi-Clique* between two context graphs determine the degree of similarity between the entities. The higher the size of the common *Quasi-Clique* found, the more similar the entities are.

Since our motivation is to reduce false positives of the existing techniques, we use our method in combination with any existing method, as seen in Figure 4.6. Any existing method may return a ranking r, we propose to use our technique to re-rank the existing ranking to boost the ranking of the correct variants in the upper rank locations.

Table 4.3. Summary of data sets. [65]

Data set	Domain	# of named 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

4.2.4 Experimental Validation

4.2.4.1 Set-up

We validate our method on one real and four synthetic data sets gathered from diverse domains: ACM, BioMed, EconPapers, and IMDB – as shown in Table 4.3. Our real case data set is from ACM and the same as the one used in Chapter 3. We prepared a syntatic data set from each of the above mentioned domains as follows. First, we choose 100 named entities with enough tuples associated. These correspond to our canonical entities whose names variant to be found. For a canonical entity name such as "Jeffrey D. Ullman", we then make up a variant name by either using the first initial ("J. D. Ullman") in the name, or injecting extra characters to the last name in 7:3 ratio, i.e., ("J. D. X-Ullman-Y"). Finally both canonical and its variant entities share half of the associated tuples (context) of the original entity.

Evaluation Metrics.

We use the traditional recall described in Chapter 3. Furthermore, we are more interested in how precise the results are for a given algorithm to see the effectiveness against the false positive problem. Since the traditional precision does not account for the quality of ranking the right variants returned, we select to use *Ranked Precision* measures precision at different cut-off points, defined as $\textit{Ranked Precision} = \frac{\sum_{i \in C} precision_i}{r}$, where $precision_i$ is the traditional precision at a

Table 4.4. Summary of notation. [65]

Notation	# Meaning
JC	Jaccard
TI	TF*IDF
IC	IntelliClean
QC	Quasi-Clique
JC + QC	Jaccard + Quasi-Clique
TI + QC	TF*IDF + Quasi-Clique
IC + QC	IntelliClean + Quasi-Clique

cut off of i [44]. Finally, the *Average Recall (AR)* and *Average Ranked Precision (ARP)* are the averages of recall and ranked precision over all cases.

Compared Methods. In order to evaluate the effectiveness of our quasi-clique based distance function, we use various traditional distance metrics in the first step to get initial candidate variants: *Jaccard (JC)*, *TFIDF (TI)* and *IntelliClean (IC)* (see [77] for definitions).

Then, we apply our *Quasi-Clique* metric to the candidate variants to get the final ranking. Then, we compare the performance of "before" and "after" *Quasi-Clique*-based metric was applied to see how much improvement we get with our scheme used along with the current methods. The summary of all the methods described in Table 4.4.

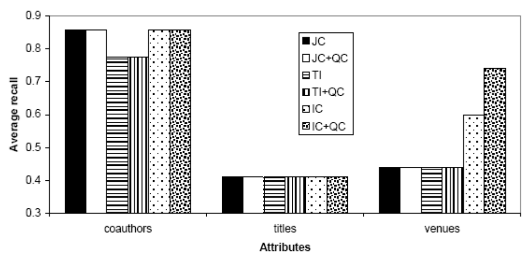
4.2.4.2 Results

For the real case from the ACM data set, the precision of the combined experiments (i.e., JC + *Quasi-Clique*) significantly improves the precision as seen in Figure 4.7(a-b). On average, the average ranked precision improves by 63%, 83%, and %46 for three attributes, respectively. The baseline methods JC, TI are based on measuring distance on common tokens, in general. Since some authors may have name initials and common first or last names on co-author data, these methods may therefore generate a large number of false positives. However, *Quasi-Clique* uses additional information that also regards the collaboration patterns, it overcomes the limitation of these simple metrics on the false positive problem.

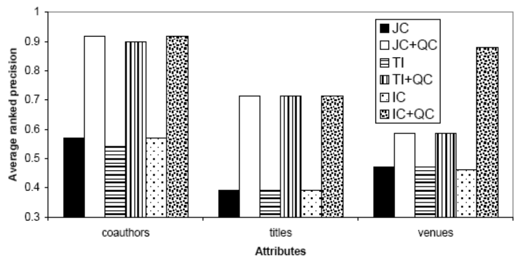
The experiments on the synthetic data sets in Figure 4.7(c-h) show the performance of our proposal against large-scale data sets. Overall, we get up to 75% in average ranked precision by our method regardless of the distance function used in the first step (JC or TI), type of the attributes used as entity context, or type of the data sets. In particular, the performance results of the IMDB data set is shown in Figure 4.7(g-h). Among the various attributes available in the data set, we chose to use "location", "production companies", and "release date" since these attributes have more contextual information than the others. The performance of the baselines JC and TI is relatively lower than the other two, yet our distQC improved the average ranked precision by 13%, suggesting its effectiveness. Our method perform sligtly better in the citation based data sets ACM and BioMed. This is because, unlike the citation cases, the records in the IMDB data set have no strong relationships, do not carry rich information and include more empty values and noises.

4.2.5 Concluding Remarks

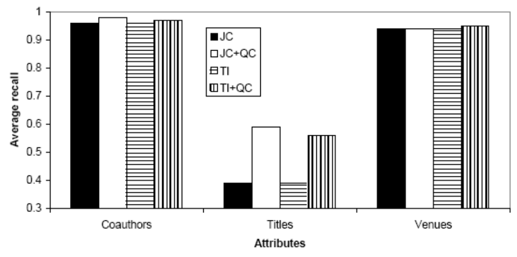
Toward the false positive problem in split name linkage, we present a graph partition based approach using the notion of *Quasi-Clique*. Unlike string distance or vector-based cosine metric, our approach examines the relationship hidden under the named entities, and use this additional knowledge to improve the precision of these textual based methods when combined together. Experimental results verify that our proposed approach improves precision up to 83% at best, but never worsens it.



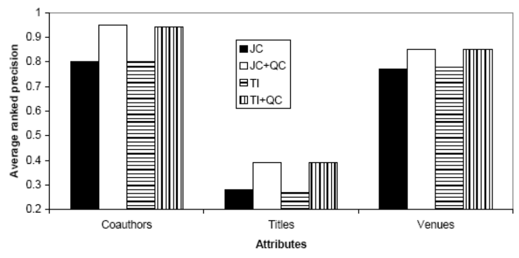
(a)



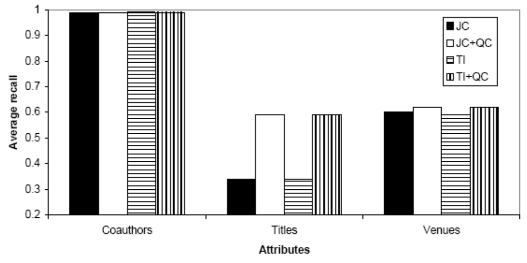
(b)



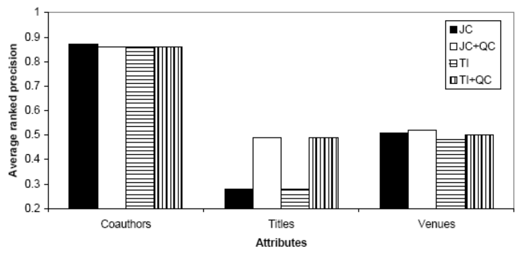
(c)



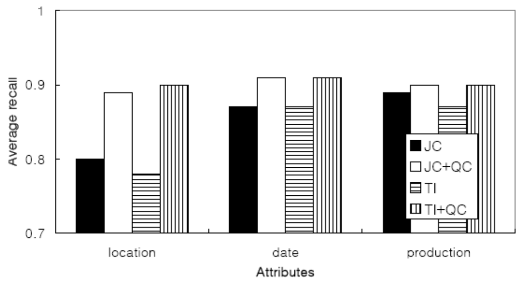
(d)



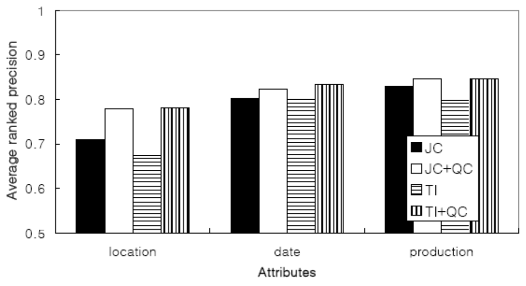
(e)



(f)



(g)



(h)

Figure 4.7. Test cases for three data sets: (a-b) real cases for ACM data set, and (c-d), (e-f), and (g-h) synthetic cases for ACM, BioMed, and IMDB data sets, respectively [65].

4.3 Modeling the Collaboration Network

In this section, we consider the problem of how the collaboration network we created from the ACM Digital Library (ACM-DL) for our linkage method can be modeled. As the data for complex networks (i.e., the WWW, power grids, social networks, food chains etc) become more readily available and having more computational power, research interest has moved to the analysis of network structure. Why are people interested in the structure of such networks? Among various reasons, it is mainly because the structure of networks always affects processes occurring in the networks. For instance, the topology of the social network of interactions may affect the spread of information or disease, and the topology of the Internet may give helpful clues about the robustness, stability or cost of communication among computers world-wide. Since most real-world networks are very large in scale, analyzing their topology may reveal important properties, without looking at the actual system or dealing with complexities associated with those systems.

The analysis of the complex networks has also drawn attention to modeling of these networks. If one could come up with a good model, the entire network can be re-generated or the future status of the network can be predicted. This is an important feature for our linkage methods. If the quality of the semantic information hidden in the social networks is higher, the better quality of the linkage output gets.

Although there are a large number of varieties, many real-world complex networks such as collaboration networks share three important characteristics [4, 26] : (1) power-law degree distribution, (2) small geodesic (i.e., average distance among nodes), and (3) high clustering coefficient. A good network model should be able to generate all these three important characteristic well.

4.3.1 Models for Collaboration Networks

There have been several network models proposed for modeling complex networks such as the AB Model [3]. The AB model is the authors' own extension to their original proposal [11]. It is a dynamic model, so one node joins to the network at each time step t and creates links to the existing nodes in the network according to an attachment rule based on the preferential attachment phenomenon described in section 4.1. In addition to these external links, internal links and rewiring of existing links are also employed. In this model three operations are available: (i) with probability p, m internal links are added. One node is selected randomly and other endpoint is selected using the attachment rule. (ii) with probability q, m links are rewired. One node, and one of the links of that node is selected randomly and using the attachment rule it is reconnected a node. (iii) with probability 1-p-q, one new node and m external links are added using the attachment rule.

Although this basic model and several other models for complex networks generate the degree distribution very well for our collaboration network from the ACM-DL (Figure 4.8), they are not sufficient to generate the other two important network characteristics: small average distance, and large clustering coefficient [29]. By extending the preferential attachment model [11], we show how to generate synthetic networks that exhibit the three properties better than this model for our collaboration network.

Since the degree distribution is correctly generated, our goal is to try to improve both average distance and clustering coefficient results. Even though absolute values may differ, it is important for a good model to regenerate those two properties at least in parallel with the actual network's results while keeping the power-law degree distribution. We try to modify the AB model which shows the best performance in terms of degree distribution [29] to achieve this goal.

4.3.2 The New Model

One problem with the existing method is that it uses fixed parameters for the probability of both internal and external link additions, no matter at what stage the network is. However, the collaboration network of the ACM-DL does not show such regularity in terms of both link types. In the initial stage where the network is relatively small and tries to enlarge, the external links dominates the network. Many new nodes compared to the initial population enter the system, and connect to the existing nodes by creating external links. In the mean time, collaboration between old nodes is uncommon and decreasing the probability for internal link additions. This situation results in less number of short cuts between existing nodes and increases the average distance over the initial years. Nevertheless, this rate does not keep up at the same level. As the network becomes more mature, the probability of existing nodes to collaborate with each other increases significantly and later catches the same rate or sometimes even larger in the recent years of the period analyzed. This is the most important factor why we see a stable decrease in the later stage of the collaboration network. Hence we can also capture this situation by not changing the model but adjusting the parameters dynamically for the probabilities of internal and external link additions. We use the AB model and compute the probability for internal link additions $p(t)$ with a linear increase as follows:

$$p(t + \delta) = p(t) + \alpha * p(t)$$

where initial internal link probability $p(0) = 0.143$ and the increase rate $\alpha = 0.15$, chosen by considering the actual collaboration network's behavior.

Another problem with the existing methods is that they employ preferential attachment mechanism globally. In other words, if a node creates a link with a preferential attachment, it can connect to any of the remaining nodes in the entire network, with some probability less or more. This is not the case seen in a collaboration network. One person usually gets new acquaintances within a limited community, i.e. a friend of a friend, another person in the same company etc. So the attachment rule works within a limited range rather than for the entire network. This creates a hierarchical structure in the collaboration network, which is responsible for high value of clustering coefficient in such networks. A collaboration, or in general, a social network is in fact fundamentally modular; one can easily identify groups or communities that are highly interconnected within the group but have only a few links to nodes outside of the group to which they belong to. Hence, there are usually many nodes that have dense connections only within their community, where as relatively small number of other nodes creates the hierarchical topology by having connections between communities. As a result, high clustering coefficients of those densely connected small groups cause the entire network to have a high average clustering coefficient [70]. Researchers have also tried to cope with the dramatic difference between the clustering coefficient values of the theoretical models and real-world networks. Among the ones that directly aim clustering coefficient issue, [42] introduces a mixture of global preferential attachment and local uniform attachment to increase the number of connected trios of nodes in the original Barabasi-Albert model [11]. The work in [70] proposes a new hierarchical network model that combines scale-free property with high degree of clustering. In their model, a network starts with a small cluster of five densely linked nodes, and then four replicas of this module are created and connected to the central node of the old cluster. The same operation recursively continues to enlarge the system to the desired size. [33] defines a higher order clustering coefficient metric for complex networks claiming that it would be more appropriate metric to give insight into the modeling of clustering mechanism rather than using standard clustering coefficient. Higher order clustering coefficient measures connectivity between a node's immediate and mode distant neighbors to a specific distance.

To get better results for clustering coefficient, we use a similar scheme as in [42]. We use the preferential attachment mechanism in all cases and limit the range for a new link addition for both internal and external linking in the AB model as follows:

1. A new node creates its first external link according to the preferential attachment. However if it creates more external links, each consecutive one must be within the 2-node neighborhood of that node with probability p, and within the entire network with probability $1 - p$.

2. An existing node creates its internal links to connect to the other old nodes according to the preferential attachment rule. However, each internal links must be within the 2-node neighborhood of that node with probability p, and within the entire network with probability $1 - p$.

4.3.3 Experimental Results

The result of this model is shown in Figure 4.8. We experimentally select $p = 0.5$. It creates the power-law degree distribution since the mechanism is still preferential

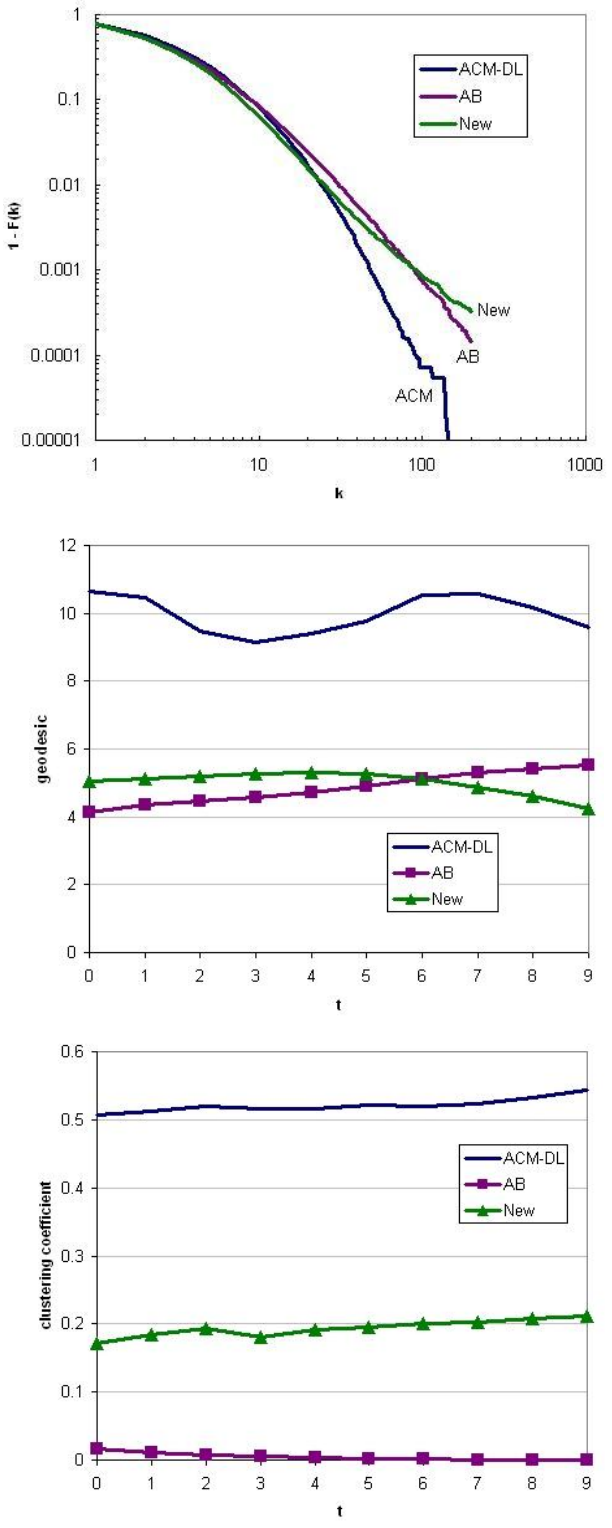


Figure 4.8. Results of the new model for the ACM-DL: (a) degree distribution, (b) average distance, (c) clustering coefficient.

attachment with a slight deviation. Moreover it achieves the similar pattern with the average distance and clustering coefficient, and generates closer absolute values to the actual collaboration network in terms of clustering coefficient performance. the AB model is also experimented for comparison, where the parameters $p = 0.455$, $m = 1.5$, without rewiring. We set $p(0) = 0.2$ and $\alpha = 0.11$ for the dynamic parameters and use $p = 0.5$ for the local preferential attachment probability. The results of the real data set, the AB model and the new model are shown in Figure 5 on the three characteristics. The AB model almost perfectly matches on the degree distribution while giving disappointing results on the remaining two metrics, like in the ACM-DL case. Despite a small deviation on the degree distribution, our new model is again able to regenerate the desired situation for the average distance and the clustering coefficient for this collaboration network as well.

Using dynamic network parameters and a mixture of local and global preferential attachment are surely two important factors in the modeling of collaboration network evolution but might not be enough as seen in the results. Many other such phenomena hidden in the actual collaboration patterns of scholars should be discovered and used along with these two factors for more accurate models. A possible future study may enrich the proposed model by considering the network to be composed of different sub networks each of which might refer a different field, study, location, etc. Each sub network may have its own events and attachment rules within the sub network and with the other sub networks, as seen in the actual collaboration networks. Incorporating more actual events happening in actual networks into a model will result in more accurate generation of such networks.

4.3.4 Concluding Remarks

Finally, we present our model that modifies the original AB model for generating a closer match and tendency for the average distance and clustering coefficient dynamism seen in our collaboration network of the authors from the ACM-DL. Using this model, the entire network can be re-generated or the future status of the network can be predicted. This could improve the quality of the semantic information hidden in the collaboration network which therefore improve the quality of the linkage process based on the collaboration network, such as our quasi-clique based linkage proposal described in the previous section.

Chapter 5

The Mixed Name Linkage Problem

In this chapter, we tackle the *Mixed Name Linkage* problem that commonly occurs in digital libraries, the Web, and many other large-scale data collections. When more than one named entity in a data collection has the exact same name spelling and there is no other identifier available, the data collection may have to list all the elements associated with those entities with the same name under that single name. Like the *Split Name Linkage* problem, this also creates dirty data for applications that use such collections, and inherits the same problems, such as poor user experience, non-accurate citation analysis etc.

As summarized in the related work in Section 2, different clustering or classification approaches are used to handle such problems in general. In this chapter, we investigate the problem from the data input perspective and figure out how most useful data pieces can be found and extracted for the effectiveness of a clustering algorithm used. We particularly focus on one challenging scenario from the Web, which we call "Web People Name Disambiguation", and show the robust performance of our scheme for this scenario in the following sections.

5.1 Web People Name Disambiguation

Web search based on person's names has been always one of the most popular query types. For instance, a recent study in [36] reports that around 30% of search engine queries include person names. Person names, however, are highly ambiguous (e.g., 90,000 distinct names are shared by 100 million people in US) [36]. Therefore, when a user searches a person name on the web, top ranked search results high likely include a mix of web pages links that belong to multiple individuals with the same name spelling. Then the user has to browse every document to find the information about the person searched for, or use more complex query that can capture the target pages better which may result in reducing recall.

The ideal system to overcome this problem receives as input a set of web pages retrieved from a search engine using a given person name as a query. The goal is to determine how many different people are represented for that name in the input web pages, and correctly assign each namesake to its corresponding subset of web pages.

There are many challenges towards an effective solution. We are to correctly estimate the number of namesakes for a given person name and group documents referring to the same individual. Moreover, the information sources to be processed are unstructured web pages and there is no certain way of correctly establishing a relation between any two web pages belonging to the same or different individuals.

As the input, we use the data set provided for the Web People Search Task 2007 [8]. This data set provides different name ambiguity problems from different resources, as follows:

- US Census. This data set is from [54] and contains 32 names randomly picked from the US Census.

- Wikipedia. Seven names were sampled from a list of ambiguous person names (popular or historical) in the English Wikipedia.

- ECDL. This data set contains ten additional names that were randomly selected from the Program Committee listing of a Computer Science conference (ECDL 2006).

All these data sets consist of collections of web pages obtained from the top-100 (upper bound) results for a person name query on Yahoo! [8].

We have taken several approaches to analyze different sources of information provided with the input data, and also compared strategies to combine these individual features together. The configuration that achieved the best performance used a single named entity feature as input to clustering. In the remainder of this chapter, we first describe our system in terms of the clustering approach used and alternative features investigated. We then analyze the results on the training set before concluding the work.

5.2 Clustering Algorithm

Clustering is the key part for such a task. We have chosen to view the problem as an unsupervised hard clustering problem. First, we view the problem as *unsupervised*, using the training data for parameter validation, to optimally tune the parameters in the clustering algorithm. Secondly, we observed that the majority of the input pages reference a single individual, although there are a few that reference multiple individuals sharing the same name. Hence, we view the problem as *hard* clustering, assigning input pages to exactly one individual, so that the produced clusters do not overlap.

Hard clustering algorithms can be classified as either partitive or hierarchical. Agglomerative hierarchical clustering generates a series of nested clusters by merging simple clusters into larger ones, while partitive methods try to find a pre-specified number of clusters that best capture the data. As the correct number of clusters is not given *a priori*, we chose a method from the second group. We use the *Hierarchical Agglomerative Clustering* (HAC) algorithm [47] for all experiments reported in this chapter. HAC views each input web page as a separate cluster and iteratively combines the most similar pair of clusters to form a new cluster that replaces the pair.

5.3 Features

As input to the clustering, we consider several different representations of the input documents. Each representation views the input web pages as a vector of features. HAC then computes the cosine similarity between the feature vectors for each pair of clusters to determine which clusters to merge. We now review the inventory of features studied in our work.

Tokens (T). Identical to the task baseline by [8], we stemmed the words in the web pages using the Porter stemmer [69], to conflate semantically similar English words with the stem. Each stemmed word is considered to be a feature and weighted by its Term Frequency $\times$ Inverse Document Frequency (TF$\times$IDF).

Named Entities (NE). We extract the named entities from the web pages using the Stanford Named Entity Recognizer [31]. This tagger identifies and labels names of places, organizations and people in the input. Each named entity token is treated as a separate feature, again weighted by TF$\times$IDF. We do not perform stemming for NE features.

We also consider a more target-centric form of the NE feature, motivated by the observation that person names can be differentiated using their middle names or titles. We first discard all named entities that do not contain any token of the search target, and then discard any token from the remaining named entities that appears in the search target. The remaining tokens are then used as features, and weighted by their TF$\times$IDF. For example, for the search target "Edward Fox", the features generated from the name "Edward Charles Morrice Fox" are "Charles" and "Morrice". We call this variation NE targeted (NE-T).

Hostnames and domains (H and D). If two web pages have links pointing to the exact same URL, then there is a good chance that these two web pages refer the same person. However, we find such exact matches of URLs are rare, so we relax the condition and consider their hostnames or domain names instead. For example, the URL http://portal.acm.org/guide.cfm has hostname portal.acm.org and domain name acm.org. As such, for each web page, we can extract the list of hostnames from the links in this page.

We observe that some host/domain names serve as more discriminative evidence than others (e.g., a link to a university homepage is more telling than a link to the list of publications page of Google Scholar when disambiguating computer science scholars). To model this, we weight each host/domain name by its IDF. Note that we do not use TF as web pages often contain multiple internal links in the form of menus or navigation bars. Using IDF and cosine similarity has been proven effective for disambiguating bibliographic citation records sharing a common author name [80].

We also considered a variant where we include the URL of the input web page itself as a "link". We tried this variation only with hostnames, calling this Host with Self URL (H-S).

Page URLs (U). Uniform resource locations (URLs) themselves contain a rich amount of information. For example, the URL http://www.cs.ualberta.ca/~lindek/ itself suggests a home page of "lindek" in the Computer Science department, University of Alberta, Canada.

We used the MeURLin system [49] to segment the URL of each web page into tokens as well as to generate additional features. These features include (a) segmentation of tokens such as "www.allposters.com" to "www", "all", "posters" and "com"; (b) the parts in the URL where the tokens occur, e.g., protocol, domain name, and directory paths; (c) length of the tokens; (d) orthographic features; (e) sequential n-grams; and (f) sequential bigrams. As each of these features can be seen as a "token", the output of the MeURLin segmenter for a web page can be seen as a "document", and hence it is possible to compute the TF×IDF cosine similarity between two such documents.

5.4 Feature Combination

The features described above represent largely orthogonal sources of information in the input: input content, hyperlinks, and source location. We hypothesize that by combining these different features we can obtain better performance. To combine these features for use with HAC, we consider simply concatenating individual feature vectors together to create a single feature vector, and compute cosine similarity. We used this method in two configurations: namely, (T + NE + H-S), (T + D + NE + NE-T + U).

We also tried using the maximum and average component-wise similarities of individual features. (max(NE, H-S)) uses the maximum value of the Named Entity and Host with Self features. For the (avg(T, H-S)) and (avg(T, D, NE, NE-T, U)) runs, we compute the average similarity over the two and five sets of individual features, respectively.

Feature	ECDL		Wikipedia		Census	
	$F_{\alpha=0.5}$	$F_{\alpha=0.2}$	$F_{\alpha=0.5}$	$F_{\alpha=0.2}$	$F_{\alpha=0.5}$	$F_{\alpha=0.2}$
Tokens (T)	.72 / .77	.83 / .84	.72 / **.76**	.85 / .84	**.82** / **.84**	.88 / .86
Named Entities (NE)	.75 / **.80**	.84 / .79	.75 / **.77**	.85 / .78	**.89** / **.78**	.89 / .73
NE targeted (NE-T)	.54 / .55	.49 / .47	.66 / .64	.60 / .57	.64 / .64	.57 / .58
Host (H)	.72 / .57	.64 / .48	.67 / .51	.58 / .41	.67 / .63	.59 / .55
Host + Self (H-S)	.73 / .59	.66 / .49	.68 / .54	.60 / .43	.68 / .63	.60 / .56
Domain (D)	**.78** / .69	.72 / .60	.71 / .59	.66 / .50	.69 / .65	.61 / .58
Domain + Self (D-S)	**.79** / .70	.74 / .61	.72 / .62	.67 / .52	.70 / .66	.62 / .59
URL (U)	.50 / .43	.43 / .35	.56 / .42	.50 / .33	.64 / .58	.56 / .51
(T + NE + H-S)	.71 / .77	.83 / .83	.72 / **.76**	.85 / .83	.65 / .67	.78 / .76
(T + D + NE + NE-T + U)	.72 / .76	.83 / .80	.72 / **.77**	.84 / .83	.66 / .66	.78 / .74
(*max*(NE, H-S))	.74 / **.80**	.84 / .82	.74 / **.77**	.86 / .82	.71 / .66	.80 / .70
(*avg*(T, H-S))	.77 / **.81**	.86 / .76	.75 / **.77**	.86 / .76	.70 / .64	.80 / .67
(*avg*(T, D, NE, NE-T, U))	**.78** / .77	.86 / .73	.75 / **.78**	.86 / .76	.69 / .61	.77 / .62

Table 5.1. Experimental results for each training data set of the task: ECDL, Wikipedia and Census. Each experiment uses single link HAC with the similarity threshold values of 0.1 / 0.2. Best $F_{\alpha=0.5}$ performances are shown in bold.

5.5 Results

We present the clustering performances of the various methods in our system based on the different features that we extracted. Each experiment uses HAC with single linkage clustering. Since the number of clusters is not known, when to terminate the agglomeration process is a crucial point and significantly affects the quality of the clustering result. We empirically determine the best similarity thresholds to be 0.1 and 0.2 for all the experiments on the three different data sets provided. We found that larger values for these data sets do not allow the HAC algorithm to create enough clustering hierarchy by causing it to terminate early, and therefore result in many small clusters increasing purity but dramatically suffering from inverse purity performance.

Table 5.1 shows the results of our experiments on the training data sets (ECDL, Wikipedia and Census). Two different evaluation measures are reported as described by the task: $F_{\alpha=0.5}$ is a harmonic mean of purity and inverse purity of the clustering result, and $F_{\alpha=0.2}$ is a version of F that gives more importance to inverse purity [7].

Among the individual features, Tokens and Named Entity features consistently show close to best performance for all training data sets. In most cases, NE is better than Tokens because some web pages contain lots of irrelevant text for this task (e.g., headers and footers, menus etc). Also, we found that the NEs have far more discriminative power than most other tokens in determining similarity between web pages. The NE variation, NE targeted, performs worse among the token based methods. Although NE targeted aims for highly precise disambiguation, it seems that it throws away too much information so that inverse purity is very much reduced. The other NEs, such as locations and organizations are also very helpful for this task. For example, the organization may indicate the affiliation of a particular name. This explains the superiority of NE over NE targeted for all three data sets.

Among the link based features, Domain gives better performance over Host as it leads to better inverse purity. The reason is that there are usually many pages on different hosts from a single domain for a given name (e.g., the web pages belonging to a researcher from university domain). This greatly helps in resolving the name while results in a slight drop in purity. Using a web page's URL itself in the features Host+Self and Domain+Self shows a larger increase in inverse purity at a smaller decrease in purity, hence these have improved F-measure in comparison to Domain and Host. Not surprisingly, these link based features perform very well for the ECDL data set, compared to the other two. A significant portion of the people in the ECDL data set are most likely present-day computer scientists, likely having extensive an web presence, which makes the task much easier. Although the other two data sets may have popular people with many web pages, their web presence are usually created by others and often scatter across many domains with little hyperlinkage between them. This explains why our link based methods are not very effective for such data sets.

Our final individual feature URL performs worst among all. Although highly precise, its resulting inverse purity is poor. While the features generated by MeURLin do improve the performance over pure host name and domain on the page URLs, its incorporation in a richer feature set does not lead to better results, as the other features which have richer information to process.

Each of the individual features has different degree of discriminative power in many different cases. By combining them, we expect to get better performance than individually. However, we do not obtain significant improvement in any of the data sets. Furthermore, in the Census data set, the combined features fail to outperform the individual NE and Tokens features. The relatively poor performance of the remaining features also degrades the performance of Tokens and NE when combined.

5.6 Concluding Remarks

We described our system that disambiguates people mentions in web pages returned by a web search scenario. As such, we mainly focus on extracting various kinds of information from web pages and utilizing them in the similarity computation of the clustering algorithm. The experimental results show that a simple Hierarchical Agglomerative Clustering approach using a single named entity feature seems promising as a robust solution for the various data sets.

Chapter 6

Conclusion

The names of entities are among the most commonly chosen identifiers for use in identifying entities. However, since those names are often non-unique and ambiguous, confusion inevitably occurs. In particular, when a variety of names are used for the same real-world entity, detecting all variants and consolidating them into a single canonical entity is a significant problem. This problem has been known as the (record) linkage or entity resolution problem. It occurs in many real world applications, such as databases, digital libraries, search engines etc. Therefore, it is crucial to have efficient solutions for the data quality of such applications and thus a better user experience.

In this thesis, we have investigated novel and efficient solutions for the linkage problem of named entities. We have separated the problem into two main cases - **Split Name Linkage** and **Mixed Name Linkage**. In the split name linkage problem, an entity and its elements may be listed under more than one name (variants). The goal is to identify all of the name variants, collapse them and place all of the elements under a single, canonical name. In the mixed name linkage problem, elements of more than one entity that have the same name spelling may be mixed together under a single name. The goal is to be able to identify the subgroups of the single group, each of which corresponds to the group of elements of a different entity with that name. While we have also proposed a solution for the mixed name linkage problem, our main focus in this thesis has been on novel and efficient solutions for the split name linkage problem.

6.1 Summary of Contributions

The major contributions of this thesis can be summarized as follows:

- We have developed a novel idea of using the Web as a source for additional knowledge of an entity to solve the split name linkage problem, particularly for cases where there are incomplete or noisy data. Since the Web is the largest source of knowledge of any kind of information, acquiring additional information about an entity becomes possible, regardless of domains or applications

- To capture the additional information of an entity returned from a search engine, we have proposed various methods, such as using the frequency of representative terms, URL information or the content of returned web pages, and created new web-based similarity measures.

- To reduce the false positive problem caused by relying solely on textual similarity, we have investigated how the entity context can be captured by a graph and how the hidden information can be mined from this context graph. We have analyzed what various types of hidden information on a context graph can be used as additional information for the split name linkage problem.

- We have presented a similarity metric on the entity context based on the notion of Quasi-Clique. As used in combination with conventional entity resolution based on textual similarity, this approach exploits the relationships in entity context and, therefore, permits conventional approaches to make stronger linkage decisions, thereby reducing false positives.

- For the mixed name linkage problem, we have investigated the problem from the data input perspective and decided how the most useful data pieces can be found and extracted for the effectiveness of the clustering algorithm used. In particular, we have selected one challenging scenario from the web that shows the robust performance of our scheme, which we call "Web People Name Disambiguation."

This research is an important step in ensuring data cleaning for large-scale data repositories. Clean input data provide better quality output from the services and applications that use the data repository and, therefore, better experience for the users of these services and applications. This may take the form of higher quality search engine results, more accurate scientific impact measurements etc.

6.2 Future Research Directions

Many directions are ahead for the future work. Our proposed solutions for the split name linkage problem can be tested on more and larger data sets.

For the web-based name linkage methods, more sophisticated approaches, such as Information gain, Chisquare, etc., can be used to find better representative tokens or key phrases from the contents of canonical entity, or both entities to be linked. It is also important to be able to quantify the signal-to-noise ratio in selecting multiple information and combining them in queries in order to acquire the best knowledge.

The Web is an evolving resource. The information available on the Web gets larger and larger, and changes over time. Web information's temporal change could be incorporated in the linkage decision. We can determine the pattern of how two given entities' collected information on the web changes over time and correlate the similarities on the patterns in a similarity metric.

Finally, both web-based and graph-based approaches proposed for the split name linkage problem could be extended to support the mixed name linkage problem, as well.

Bibliography

[1] ACM. The ACM digital library.

[2] E. Agirre, O. Ansa, E. Hovy, and D. Martinez. "Enriching Very Large Ontologies Using the WWW". In *ECAI Ontology Learning Workshop*, Berlin, Germany, 2000.

[3] Reka Albert and Albert-Laszlo Barabasi. Topology of evolving networks: local events and universality. *Physical Review Letters*, 85:5234, 2000.

[4] Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

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

[6] Hiroko Ao and Toshihisa Takagi. ALICE: An algorithm to extract abbreviations from MEDLINE. *Journal of the American Medical Informatics Association*, 12(5):576–586, September/October 2005.

[7] Javier Artiles, Julio Gonzalo, and Satoshi Sekine. The SemEval-2007 WePS evaluation: Establishing a benchmark for the Web People Search Task. In *SemEval 2007, ACL*, June 2007.

[8] Javier Artiles, Julio Gonzalo, and Felisa Verdejo. A testbed for people searching strategies in the WWW. In *ACM SIGIR*, pages 569–570, August 2005.

[9] A. L. Barabasi and R. Albert. "Emergence of Scaling in Random Networks". *Science*, 286(509–512), 1999.

[10] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. "Evolution of the social network of scientific collaborations". *Physica*, A 311:590–614, 2002.

[11] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[12] V. Batagelj and A. Mrvar. "Pajek - A program for large network analysis". *Connections*, 21(2):45–57, 1998. http://vlado.fmf.uni-lj.si/pub/networks/pajek/.

[13] Ron Bekkerman and Andrew McCallum. Disambiguating web appearances of people in a social network. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 463–470, New York, NY, USA, 2005. ACM.

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

[15] 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.

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

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". *Journal of Machine Learning Research*, (3):993–1022, May 2003.

[18] Jeffrey T. Chang, Hinrich Schutze, and Russ B. Altman. Creating an online dictionary of abbreviations from MEDLINE. *Journal of the American Medical Informatics Association*, 9(6):612–620, November/December 2002.

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

[20] R. Cilibrasi and P. M. B. Vitanyi. "Automatic Meaning Discovery Using Google", 2004.

[21] P. Cimiano, S. Handschuh, and S. Staab. "Towards the Self-annotating Web". In *Int'l World Wide Web Conf. (WWW)*, pages 462–471, 2004.

[22] 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.

[23] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, March 2000.

[24] DBLP. The DBLP computer science bibliography.

[25] X. Dong, A. Y. Halevy, and J. Madhavan. "Reference Reconciliation in Complex Information Spaces". In *ACM SIGMOD*, 2005.

[26] S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks. *Advances in Physics*, 51:1079, 2002.

[27] E. Elmacioglu and D. Lee. "On Six Degrees of Separation in DBLP-DB and More". *ACM SIGMOD Record*, 34(2):33–40, 2005.

[28] E. Elmacioglu and D. Lee. "De-duplication by Googling". Technical report, Penn State University, 2006.

[29] Ergin Elmacioglu and Dongwon Lee. "Modeling Idiosyncratic Properties of Collaboration Networks Revisited". *Scientometrics (forthcoming).*

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

[31] Jenny R. Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*, pages 363–370, June 2005.

[32] L. C. Freeman. "A Set of Measures of Centrality Based on Betweenness". *Sociometry*, 40(35–41), 1977.

[33] Agata Fronczak, Janusz A. Holyst, Maciej Jedynak, and Julian Sienkiewicz. Higher order clustering coefficients in barabasi-albert networks. *PHYSICA A*, 316:688, 2002.

[34] G. Geleijnse and J. Korst. Automatic ontology population by googling. In *In Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2005), pages 120 – 126, Brussels, Belgium.*, 2005.

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

[36] R. Guha and A. Garg. Disambiguating people in search. Technical report, TAP: Building the Semantic Web, Stanford University, 2004.

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

[38] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 296–305, New York, NY, USA, 2004. ACM.

[39] Hui Han, Hongyuan Zha, and C. Lee Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 334–343, New York, NY, USA, 2005. ACM.

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

[41] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase: a repository of Web pages. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):277–293, 2000.

[42] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65:026107, 2002.

[43] Y. Hong, B.-W. On, and D. Lee. "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach". In *European Conf. on Digital Libraries (ECDL)*, Bath, UK, Sep. 2004.

[44] D. Hull. "Using Statistical Testing in the Evaluation of Retrieval Experiments". In *ACM SIGIR*, pages 329–338, Pittsburgh, 1993.

[45] David Hull. Using statistical testing in the evaluation of retrieval experiments. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338, June/July 1993.

[46] Panagiotis G. Ipeirotis, Eugene Agichtein, Pranay Jain, and Luis Gravano. To search or to crawl? Towards a query optimizer for text-centric tasks. In *ACM SIGMOD International Conference on Management of Data*, pages 265–276, June 2006.

[47] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

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

[49] Min-Yen Kan and Hoang Oanh Nguyen Thi. Fast webpage classification using URL features. In *CIKM*, pages 325–326, October/November 2005.

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

[51] 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)*, Jun. 2005.

[52] A. J. Lotka. “The frequency distribution of scientific production”. *J. Walsh Acad. Sci.*, 16:317–323, 1926.

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

[54] Gideon S. Mann. *Multi-document statistical fact extraction and fusion*. PhD thesis, Johns Hopkins University, Baltimore, MD, USA, 2006. Adviser-David Yarowsky.

[55] K. Markert, N. Modjeska, and M. Nissim. “Using the Web for Nominal Anaphora Resolution”. In *EACL Workshop on the Computational Treatment of Anaphora*, 2003.

[56] Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. Polyphonet: an advanced social network extraction system from the web. In *WWW*, pages 397–406, 2006.

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

[58] F. Menczer. “Growing and navigating the small world Web by local content”. *Proc. Natl. Acad. Sci.*, 99(14014–14019), 2002.

[59] S. Milgram. “The Small World Problem”. *Psychology Today*, 2:60–67, 1967.

[60] M. A. Nascimento, J. Sander, and J. Pound. “Analysis of SIGMOD’s Co-Authorship Graph”. *ACM SIGMOD Record*, 32(3), Sep. 2003.

[61] M. E. J. Newman. “The Structure of Scientific Collaboration Networks”. *Proc. Natl. Acad. Sci.*, 98:404–409, 2001.

[62] Kenneth W. Ng, Zhenghao Wang, Richard R. Muntz, and Silvia Nittel. Dynamic query re-optimization. In *International Conference on Statistical and Scientific Database Management (SSDBM)*, pages 264–273, July 1999.

[63] B.-W. On, D. Lee, J. Kang, and P. Mitra. “Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework”. In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2005.

[64] Byung-Won On. *Data Cleaning Techniques By Means of Entity Resolution*. PhD thesis, The Pennsylvania State University, University Park, PA, USA, 2007. Adviser-Dongwon Lee.

[65] Byung-Won On, Ergin Elmacioglu, Dongwon Lee, Jaewoo Kang, and Jian Pei. Improving grouped-entity resolution using quasi-cliques. In *ICDM ’06: Proceedings of the Sixth International Conference on Data Mining*, pages 1008–1015, Washington, DC, USA, 2006. IEEE Computer Society.

[66] Youngja Park and Roy J. Byrd. Hybrid text mining for finding abbreviations and their definitions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 126–133, June 2001.

[67] 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.

[68] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *KDD ’05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 228–238, New York, NY, USA, 2005. ACM.

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

[70] Erzsebet Ravasz and Albert-Laszlo Barabasi. Hierarchical organization in complex networks. *Physical Review E*, 67:026112, 2003.

[71] Bill Rosenblatt. “The Digital Object Identifier: Solving the Dilemma of Copyright Protection Online”. *Ann Arbor, MI: Scholarly Publishing Office, University of Michigan, University Library*, 3(2), December 1997.

[72] R. Sætre, A. Tveit, T. S. Steigedal, and A. Lægreid. Semantic annotation of biomedical literature using google. In *ICCSA (3)*, pages 327–337, 2005.

[73] G. Salton and M. McGill. *“Introduction to Modern Information Retrieval”*. McGraw-Hill, 1983.

[74] S. Sarawagi and A. Bhamidipaty. “Interactive Deduplication using Active Learning”. In *ACM KDD*, 2002.

[75] Ariel S. Schwartz and Marti A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing (PSB)*, pages 451–462, January 2003.

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

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

[78] Y. F. Tan, M.-Y. Kan, and D. Lee. "Search Engine Driven Author Disambiguation". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2006.

[79] Yee Fan Tan, Ergin Elmacioglu, Min-Yen Kan, and Dongwon Lee. Efficient web-based linkage of short to long forms. In *WebDB: Proceedings of the 11th annual ACM international workshop on Web information and data management*, 2008.

[80] Yee Fan Tan, Min-Yen Kan, and Dongwon Lee. Search engine driven author disambiguation. In *ACM/IEEE JCDL*, pages 314–315, June 2006.

[81] Peter D. Turney. Mining the Web for synonyms: PMI–IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167, 2001.

[82] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., 1995.

[83] J. W. Warnner and E. W. Brown. "Automated Name Authority Control". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, 2001.

[84] S. Wasserman and K. Faust. *"Social Network Analysis"*. Cambridge University Press, 1994.

[85] D. J. Watts and S. H. Strogatz. "Collective dynamics of 'small-world' networks". *Nature*, 393(440–442), 1998.

[86] Wikipedia.

[87] Y. Zhang, J. Callan, and T. Minka. "Novelty and redundancy detection in adaptive filtering". In *Proc. ACM SIGIR 2002, pp.81-88.*, 2002.

[88] Yali Zhu, Elke A. Rundensteiner, and George T. Heineman. Dynamic plan migration for continuous queries over data streams. In *ACM SIGMOD International Conference on Management of Data*, pages 431–442, June 2004.

Vita

Ergin Elmacioglu

Ergin Elmacioglu is a Ph.D. candidate and a graduate researcher in the Department of Computer Science and Engineering, the Pennsylvania State University, University Park. He received his B.S. degree from the Department of Computer Science and Engineering, Marmara University, Istanbul, Turkey, in 2001. Then, he enrolled in the Ph. D. program in the Department of Computer Science and Engineering at the Pennsylvania State University, University Park. He is a member of ACM and served as a technical referee for numerous journals and conferences.