FINDING EMAIL CORRESPONDENTS IN SOCIAL NETWORKS

by

Yi Cui
B.Eng., Shanghai Jiao Tong University, 2009

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2011

APPROVAL

Name: Yi Cui

Degree: Master of Science

Title of Thesis: Finding Email Correspondents in Social Networks

Examining Committee: Dr. Jiangchuan Liu,
Associate Professor, Computing Science
Simon Fraser University
Chair

Dr. Jian Pei,
Associate Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Andrei A.Bulatov,
Associate Professor, Computing Science
Simon Fraser University
Supervisor

Dr. Wo-Shun Luk,
Professor, Computing Science
Simon Fraser University
Examiner

Date Approved: 9 August 2011

Declaration of
Partial Copyright Licence

Abstract

Email correspondents play an important role in many people's social networks. Finding email correspondents in social networks accurately, though may seem to be straightforward at a first glance, is challenging. To the best of our knowledge, this problem has not been carefully and thoroughly addressed in research. Most of the existing online social networking sites recommend possible matches by comparing the information of email accounts and social network profiles, such as display names and email addresses. However, as shown empirically in this thesis, such methods may not be effective in practice.

In this thesis, we systematically investigate the problem and develop a practical data mining approach. Our method not only utilizes the similarity between email accounts and social network user profiles, but also explores the similarity between the email communication network and the social network under investigation. We demonstrate the effectiveness of our method using two real data sets on emails and Facebook.

To my parents and Yan

"Science without religion is lame, religion without science is blind."

— Albert Einstein, 1879 - 1955

Acknowledgments

Foremost, I would like to convey my special gratitude to my senior supervisor, Dr. Jian Pei, for his guidance, support, and supervision throughout my master's program. He gave me invaluable insights that shaped my ideas and approaches to my research. His detailed and constructive suggestions were indispensable in development of the conducted research and the thesis.

I would like to express my sincere thanks to Dr. Andrei A.Bulatov for being my supervisor and Dr. Wo-Shun Luk for being my examiner and taking the time to review the thesis. I also would like to express my sincere thanks to Dr. Jiangchuan Liu for chairing my thesis defense.

Additionally, I would thank all my friends, lab mates, classmates, faculty and support staff in our school.

Last but not least, I truly appreciate the support of my parents and my girl friend Yan Ding for their supporting and understanding all the time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

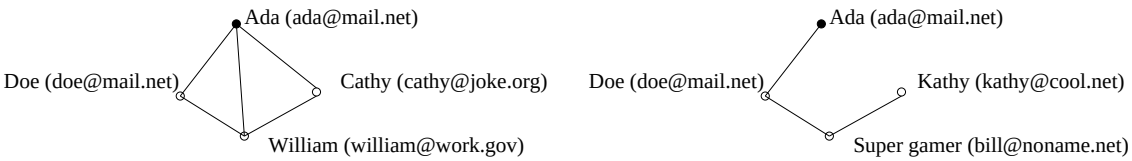
1.1 Motivations

Many people use emails everyday. Emails and the social network formed by email correspondents play an important role in many people's social life. Therefore, many online social networks, such as Facebook, LinkedIn, and Twitter, associate users with their emails in one way or another. For example, Facebook uses email addresses as user-ids. Moreover, some online social networks are trying to integrate multiple messaging channels, including SMS, chat, email, and messages. Users can send and receive messages through whatever medium or device preferred by or convenient to them. Recently, Facebook is providing an `@facebook.com` email address to every user so that a user can share with her friends over email no matter they are on Facebook or not.

One interesting and important task is to find a user's email correspondents in a target social network, such as Facebook.

Example 1 (Motivation) *Figure 1.1(a) shows some of Ada's email contacts. Two contacts are linked by an edge if they two are involved together in an email. Ada is a member of a social network, shown in Figure 1.1(b). The problem addressed in this thesis is how Ada can find her email contacts in the social network.*

Ada may add Doe as her friend in the social network, and thus an edge is added in Figure 1.1. Suppose Ada does not know her other email contacts' account information in the social network. How can she find her other email contacts, such as Cathy and William, in the social network? ■



(a) Some of Ada's email contacts

(b) Some of Ada's friends in a social network

Figure 1.1: A motivation example, where the accounts Cathy and Kathy, William and Super gamer, respectively, are owned by the same persons.

The task of finding a user's email correspondents in a social network may seem to be easy at a first glance. A straightforward solution is to search the user profiles in a social network using email account information, such as display names and email addresses. Many online social networking sites provide such functionalities, such as "find friends" in Facebook and "search for someone by name" in LinkedIn.

Example 2 (Motivation cont'd) *In our motivation example (Figure 1.1), Ada may use the user profile search function of the social network to search for her email contacts. For example, by searching for "Cathy", a good search function may recommend "Kathy" as a possible match since the two names are very similar. However, the function may not be able to match the account "Super gamer" in the social network with the email contact "William".*

■

As many experienced social network users may know, matching only based on profiles is far from enough to satisfactorily solve the problem of finding email corespondents in social networks. There are at least two major difficulties.

First, a contact may use different email addresses in email communication and social networks. Many users may have multiple email addresses. One may use a private email address for most of the email communication, and use another public email address, such as one from a free Web-based email service (e.g., `hotmail` and `gmail`), as her public email address. In such a case, searching using a contact's private email address cannot find her correctly in a social network where she registers using her public email address.

Second, one may think searching using names is reliable. However, a popular name may be used by many people. Moreover, one may not use her real name in a social network. Instead, she may use her nickname to register in a social network, or even use multiple nicknames for multiple accounts. Consequently, the results from searching using names,

though providing some candidate matches, may still contain much ambiguity and need resolution.

Although Facebook once provided a contact import functionality for Gmail users (which was disabled due to data reciprocity issue between Google and Facebook), the importer only did simple searching. If one contact's email address was found in existing Facebook user-ids, the importer gave the Gmail user a friend recommendation, otherwise sent an email to invite the contact to join Facebook. It did not utilize any linkage information. Moreover, the contact owner may have already joined Facebook with its public email address and the invitation email turns out to be annoying.

The task of finding email correspondents in social networks is important not only for individual users but also for social networking sites. First, it is a critical functionality to attract users. If a user can easily find her correspondents in a social network, she may be better committed into the social network, and more communication traffic between her and her friends may be migrated to the social network. Second, it is a critical functionality to integrate multiple messaging channels. This is particularly important for both social networking and email service providers. For example, there are some add-on applications that can let Microsoft Outlook users to keep connected with their email correspondents' Facebook statuses. Finding email correspondents in social networks effectively can help the integration of multiple messaging channels significantly. Last but not least, finding email correspondents in a social network is an instance of mapping users in two social networks, since the email communication network itself indeed is a social network. This is an interesting and challenging problem for many social network service providers, since an effective solution to this problem definitely helps those service providers to gain more users and communication traffic volume.

1.2 Technical Challenges

The main technical challenge of this thesis is the privacy issues. To work on the problem of finding email correspondents in social networks, data from both emails and social networks should be ready before algorithms go. Two volunteers provide us their data from emails and social networks for testing, whose details are in Section 6.1.

For data from emails, volunteers may concern about their information disclosure. They do not want to completely share their emails, which is understandable. We have to compile

program, deliver the program to them, and ask them to run on their machine to preserve email privacy. In this case, it is nearly impossible to analyze the data in a *white box*. Moreover, programs have to be built being compatible with different platforms, since volunteers may prefer different operating systems.

For data from social networks, the online social network websites have certain terms of use to prohibit crawling unless you apply and get a permission from them. However, it is very hard to get a permission due to the company policies and their protections on customers' privacy as well. In addition, there is no available APIs to retrieve what you want for this work, such as a given user's friends lists. For this thesis, we build our own crawler which can download and parse over $400,000$ pages per day.

1.3 Contributions

The problem of finding email correspondents in social networks has not been carefully and thoroughly addressed in research. In this thesis, we tackle the problem from a practical data mining angle, and make several contributions:

1. We model the problem of finding email correspondents in social networks.
2. We develop a practical method, which not only utilizes the similarity between email accounts and social network user profiles, but also explores the similarity between the email communication network and the social network.
3. We design a graph vertices mapping algorithms for both directed and undirected graphs, which significantly outperforms other previously known graph matching algorithms.
4. We evaluate our methods using real data sets. Empirically, we show that only when both the profile similarity and graph similarity are considered, good accuracy can be obtained.

1.4 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we formulate the problem and present our solution framework. Chapter 3, we review the previous studies related to ours

briefly. We discuss the profile similarity in Chapter 4, and develop the neighborhood based similarity and the computation methods in Chapter 5. We report an empirical evaluation in Chapter 6, and conclude the thesis in Chapter 7.

Chapter 2

Background and Problem Framework

In this chapter, we first outlines the background for this thesis which contains graph definitions and matching problems formulation. Then, we formulate the problem of finding email correspondents in social networks. Finally, we decompose the problem and present our solution framework.

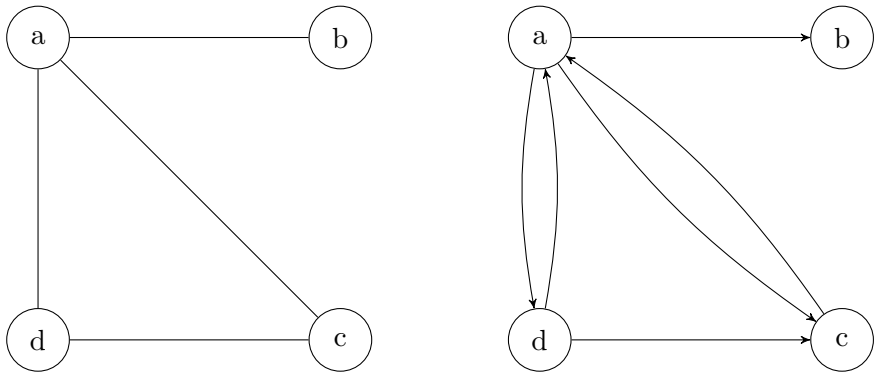
2.1 Graphs

In mathematics and computer science, a *graph* $G = (V, E)$ is an abstract structure of a set of vertices V and edges E ($E \subseteq V \times V$), where every edge connects two vertices as a representation of linkage between vertices. To be specific, the set of vertices is on behalf of a set of objects from a certain collection while the set of edges reflects pairwise relations in objects.

2.1.1 Graph Preliminaries

For an edge (u, v) where $(u, v) \in E$, the *source* of the edge is vertex u and the *target* of the edge is vertex v. The edges could be either *directed* or *undirected*. If the edge (u, v) is undirected, edge (v, u) also exists in the graph. For one graph, if it only contains undirected edges, it is *undirected graph*; otherwise, it is *directed graph*. To be specific, if one graph contains both undirected and directed edges, undirected edges (u, v) can be replaced by two

directed edges (u, v) and (v, u) to preserve directed characteristic. Two example graphs are shown as blow in Figure 2.1.



(a) An example undirected graph G_{un}

(b) An example directed graph G_{di}

Figure 2.1: Two example graphs.

2.1.2 Neighbors of Vertices

In undirected graph, a vertex v is the *neighbor* of u if there is an edge between u and v. In directed graph, neighbors of a vertex split into *in-neighbors* and *out-neighbors* according to their directions. An edge (v, u) directed from v to u introduces an *in-vertex* for vertex u. The in-neighbors of vertex u are the collections of all in-vertices of u. Similarly, we can define *out-vertex* and *out-neighbors*. In Figure 2.1, the neighbors of every vertex of each example are shown as following tables:

Vertex	Neighbors
a	$b,\ c,\ d$
b	a
c	$a,\ d$
d	$a,\ c$

Table 2.1: Neighbors of every vertex in G_{un}

Vertex	In-neighbors	Out-neighbors
a	$c,\ d$	$b,\ c,\ d$
b	a	$\emptyset$
c	a	$a,\ d$
d	$a,\ c$	a

Table 2.2: Neighbors of every vertex in G_{di}

2.1.3 Adjacency Matrices

To represent a graph, in contrast to list all vertices and edges, a convenient and easy way is to use *adjacency matrix*. For a graph $G = (V, E)$, if the cardinality of V is n, the graph's adjacency matrix M is an $n \times n$ matrix in which entry $m_{ij} = 1$ if and only if the edge (i, j) exists, $m_{ij} = 0$ otherwise. Take G_{un} and G_{di} as examples again, we could get their adjacency matrices M_{un} and M_{di} as follows:

$$M_{un} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{2.1}$$

$$M_{di} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

2.1.4 Weighted Graphs

A *weighted graph* is a graph whose edges are assigned to weighted values. In contrast to unweighted graph, whose adjacency matrix is filled with 0 and 1, weighted graphs have weighted adjacency matrices with various values. For a given weighted graph, the entry of its adjacency matrix m_{ij} is equal to the weight of the edge (i, j).

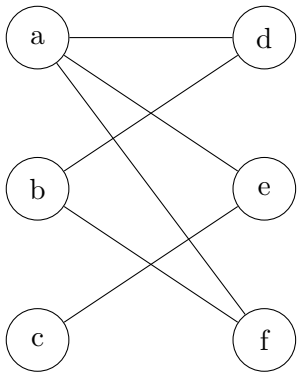
Rather than weighted-edge graph, it is also possible to have weighted-vertices in graph. However, the weighted vertices is beyond the discussion of this thesis.

2.1.5 Bipartite Graphs

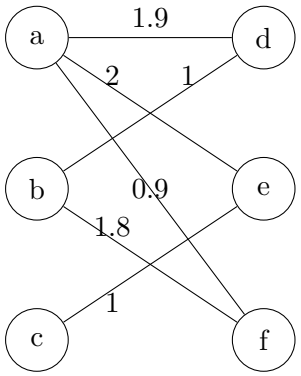
A *bipartite graph* is a graph whose vertices are composed of two disjoint vertex sets such that there are no inner-connected edges in the graph but only inter-connected edges. In other words, all edges in bipartite graph connect one vertex of one set to another vertex of the other set. Figure 2.2(a) shows an example of a bipartite graph G_{bi}, where no edges exist inside of the disjoint vertex set $\{a, b, c\}$ or $\{d, e, f\}$.

For a given graph, a *matching* is a collection of edges such that there are not any two edges sharing a common vertex. For instance, a matching of G_{bi} in Figure 2.2(a) can be $\{(a, f), (b, d), (c, e)\}$, $\{(a, d), (c, e)\}$, etc. In this thesis, we restrict our matching problem to bipartite matching problem.

If the given graph is a weighted bipartite graph, one of the most canonical problems in graph theory is *maximum weighted bipartite matching* problem. This problem is to find maximum weighted matching in bipartite graph, which is defined as the matching with the highest sum of weight of edges. For example, the maximum weighted bipartite matching in Figure 2.2(b) is $\{(a, e), (b, f)\}$ with weight sum 3.8.



(a) An example bipartite graph G_{bi}



(b) An example weighted bipartite graph G_{wb}

Figure 2.2: Two example bipartite graphs.

2.2 Problem Formulation

We consider email communication. For each email account, we assume that there is a **profile**. Typically, the profile of an email account in practice contains the email address and optionally the display name. The email communication can be modeled as an **email network** $G_M(V_M, E_M)$, where V_M is a set of email accounts, and for two accounts u and v, $(u, v) \in E_M$ is an edge if u and v are involved together in at least one email. Here, an email account u is involved in an email if u is appears in the from, to, or cc fields of the email. For the sake of simplicity, we consider an email network only as an undirected, unweighted, simple graph in this thesis.

For an account $u \in V_M$, an account $v \in V_M$ is called an **email correspondent**, or a **contact**, of u if $(u, v) \in E_M$. We denote by $C_u = \{v|(u, v) \in E_M\}$ the **set of u's contacts**.

We also consider a **social network** $G_N = (V_N, E_N)$ among people, where V_N is a set of accounts in the network, and E_N is a set of edges between accounts. Again, for the sake of simplicity, we assume that G_N is an undirected, unweighted, simple graph. Each account in the social network also has a profile, which contains information like name, email address, gender, and location. To keep our discussion simple, we assume that each person has at most one account in a social network. We will discuss how this assumption can be removed easily in Section 7.

The people in the email graph and those in the social network may overlap. That is, some people may participate in both networks. Formally, we assume that there exists a **ground truth network** $G = (V, E)$, which captures all relationships among all people. One person has one and only one vertex in the ground truth graph.

Thus, there exists a mapping $f : V_M \to V$ from the email accounts to their owners, such that for any email accounts $u, v \in V_M$, if $(u, v) \in E_M$, then either $(f(u), f(v)) \in E$, i.e., u and v are connected in the ground truth network, or $f(u) = f(v)$, i.e., u and v belong to the same person. The mapping f in general is many-to-one, since one person may have multiple email addresses.

Analogously, there exists a mapping $g : V_N \to V$ from the social network accounts to their owners, such that for any social network accounts $u, v \in V_N$, if $(u, v) \in E_N$, then either $(f(u), f(v)) \in E$ or $f(u) = f(v)$. It may not be one-to-one, either, since one person may have multiple social network accounts. In general, the ground truth graph G and the mappings f and g may not be known.

The problem of **finding email correspondents in social networks** is to find a mapping $h : V_M \rightarrow V_N$ such that for any $u \in V_M$ and $v \in V_N$, $h(u) = v$ if $f(u) = g(v)$. That is, h maps an email account u to a social network account v if both u and v belong to the same person. Technically, we define the mapping from V_M to V_N because the owner of a social network account may have more than one email account.

To tackle the problem of finding email correspondents in social networks, we need to assume the email network and the social network being available. However, in many cases this assumption cannot be met due to the privacy preservation constraint. To make our study practical, we focus on the **personal view version** of the problem, which only finds the mapping h for the set of contacts C_u for a given email account u. By focusing on the personal view version of the problem, we only have to assume that all emails involving a given account are available, and only the neighbors of a social network account are searched. This is practically achievable. The personal view version of the problem does not share the emails of an account with any other party, and can be run on the user side. Thus, the privacy of the email account is not breached.

A solution to the personal view version of the problem can be directly employed by email client tools like Outlook, or email service providers like hotmail and gmail. We also assume that our method can crawl the neighborhood of an social network user, or part of it. This assumption is practical since many social networks do allow such crawling in one way or another.

Straightforwardly, our method can be extended to tackle the general problem of finding email correspondents in social networks, such as the situation where a social network and an email provider have some business agreement in place. We will discuss this issue in Section 7.

In this thesis, instead of computing the mapping h directly, we will develop a top-k recommendation solution. That is, for each email contact u, we provide up to k social network accounts that are most likely owned by the email address owner, where k is a user-specified parameter. This design decision responds to the need and practice in existing social networking sites where more often than not a user is offered a list of recommendations.

2.3 The Framework of Our Solution

As discussed in Section 2.2, we have two types of information in finding email correspondents.

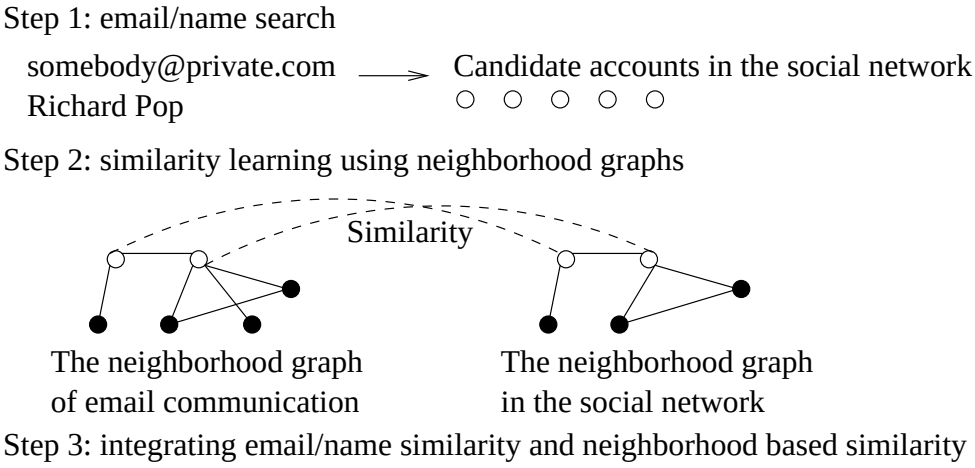


Figure 2.3: The framework of our approach.

Profile information. We have the profiles in both the email network and the social network. Therefore, we can match the email accounts and social network accounts according to the profile information. We call this the **profile matching problem**, which will be discussed in Chapter 4.

Graph information. We have the email network and the social network themselves as graphs. Heuristically, if two persons communicate well by email, they may have a good chance to be connected in a social network. Thus, we can compare the email graph and the social network graph to identify possible matching. We call this the **graph matching problem**, which will be discussed in Chapter 5.

Overall, we should integrate the results obtained from the two aspects. Thus, the framework of our method has two steps, as illustrated in Figure 2.3.

1. **Profile-based similarity search.** For each contact whose social network account u to be found, we search the social network using u's email account profile information, such as the email address and display name. Here, we assume that the social network G_N provides a search function. This step is built directly on the existing services available in many social networks. For each candidate returned by the social network search function, we calculate a similarity score between the contact and the possible match, which is called the *profile similarity*.

2. **Graph-based similarity search.** We extract the email communication graph of the contacts and the neighborhood subgraph of the possible matches obtained in the first step. Then, we compare the two graphs and derive a *graph similarity* between a contact and each possible match.

3. **Integration.** In the last step, we integrate the two similarity scores and recommend for each contact a ranked list of possible matches in the social network.

Chapter 3

Related Work

In general, our study is related to the existing work on email mining, string similarity measures, and graph matching. Limited by space, we only review the related work briefly in this section. A thorough review is far beyond the capacity of this thesis.

3.1 Email Mining

Email is one of the most popular communication ways nowadays. Emails and the email communication networks carry both the text messages people pass on to each other and the implicit social information, such as the email account owner's friends and the topics they often discussed.

Many techniques have been used in email mining. For example, Pal [24] and Carvalho *et al.* [6] applied clustering, natural language processing and text mining techniques to group email recipients. Balamurugan *et al.* [2] and Sahami *et al.* [27] used classification methods to detect spam emails.

Many applications have been developed based on email mining. For example, Roth *et al.* [26] suggested friends based on an implicit social graph built on email senders, receivers and interactions. Their study led to two interesting Gmail Labs features. The first one suggests to a user contacts for the "`To:`" field once she inputs more than two receivers in that field. The other function detects whether there is any current recipients entered by the user can be replaced by a more related contact. As another example, in the context of a "personal email social network" on people's email accounts, Yoo *et al.* [34] tackled the email overloading problem using the importance of email messages according to the

email senders' priorities. A sender's priority is calculated based on three features: the social clusters that the sender belongs to in the social network, the social importance that is the sender's centrality level in the social network, and the importance propagation level in the range from 1 to 5.

Most of the previous studies on email mining focus on emails themselves, such as mining email elements like senders, receivers, and textual contents. Different from those studies, our study is on a novel problem, mapping email correspondents and social network accounts.

3.2 String Similarity Measure

Measuring similarity between strings is a well studied problem and a widely used technique in information retrieval. Many methods have been proposed to capture the similarities between strings. For example, the Hamming distance [14] counts the total number of different characters between two equal length strings. The Jaccard similarity coefficient [25] is given by the size of the intersection over the size of the union of two sets. It can be extended to strings. The Dice similarity [10] and the overlap similarity [22] are related to the Jaccard similarity coefficient.

Many applications use one measure or a combination of multiple measures to decide the similarity between strings or documents. For example, Michelson *et al.* [23] employed a score that combines the Jaro-Winkler similarity [33], the Jaccard similarity [25] and some others to determine the best candidate from a collection of reference sets matching a post that is essentially a piece of text. Cohen *et al.* [8] compared the performance of different string similarities and their possible combinations for the task of matching names and records, including the Jaccard similarity, the Jaro-Winkler similarity, the Jaro similarity [18], and some others. Elmagarmid *et al.* [12] surveyed different types of string similarity measures on strings when they are applied to duplicate record detection.

In our study, we adopt two similarity measures for strings based on the nature of our problem. We use their combinations in matching profiles.

3.3 Graph Matching

As a fundamental problem in pattern recognition, graph matching [5] has numerous applications in various areas, such as web search, semantic networks, computer vision, and

biological networks. There are a wide spectrum of graph matching algorithms with different characteristics. Bunke [4] presented a systematic survey.

Cordella *et al.* [9] and Ullmann *et al.* [30] proposed subgraph matching algorithms based on tree search. Almohamad *et al.* [1] suggested a linear programming approach to the weighted graph matching problem. Van Wyk *et al.* [31] presented a graph matching algorithm from the functional interpolation theory point of view.

3.3.1 HITS and Iteration Framework

Kleinberg *et al.* [19] used an iterative method to update hub and authority scores to identifies good authorities and hubs of a topic for web searching which provided a general framework for iterative computing scores of vertices.

The motivation of Kleinberg's *Hypertext Induced Topic Selection* (HITS) is to find the high quality search results for queries. Traditional text-based search engines rank pages based on the query's appearing frequency which always fail to find most relevant pages for a given query. Consider the query **SFU**, its most authoritative page should be `www.sfu.ca`. However, the page of `www.sfu.ca` is not the one that uses **SFU** most often, and may not be favored by traditional text-based search engines.

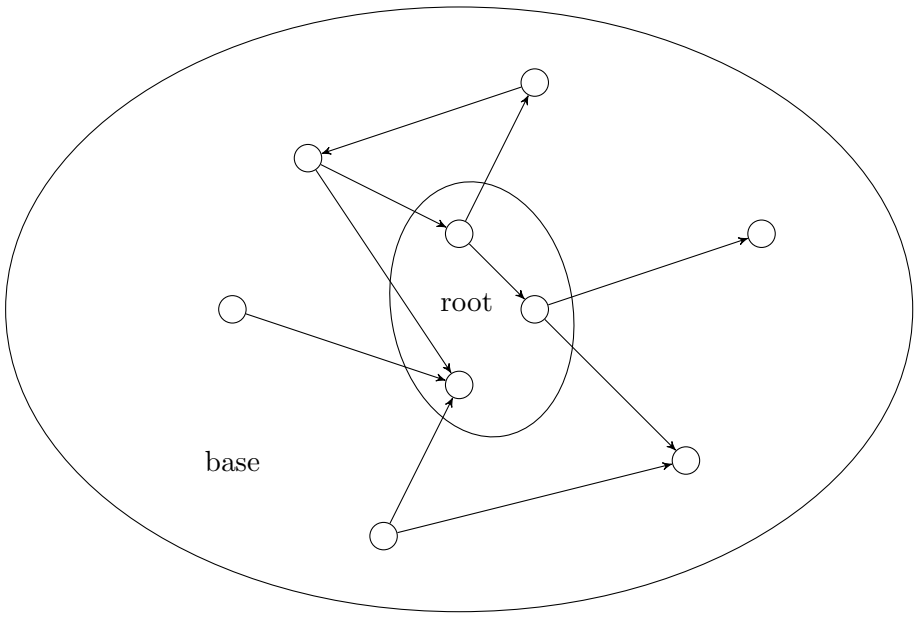


Figure 3.1: Expanding the *root* page set into a *base* page set [19].

Instead of global approaches which involve all pages and may contain millions of pages, Kleinberg proposed a *local* approach that only contain relatively small size of pages which are however rich in relevant pages. To get the *base* pages set, a collection of pages for authority and hub scores computation, Kleinberg collect the text-based search engine's top-t highest ranking pages for the query σ, which compose the page set *root* as shown in Figure 3.1 (figure is from [19]). Then a number of neighbors of every page in root page set is fetched and added into base page set. The workflow of expanding the root pages set to the base page set is shown in Algorithm 1 [19]:

Algorithm 1 GET_BASE (σ,t,d)

σ : a query
t, d : natural number parameters
$R_\sigma \leftarrow$ the top-t results of the text-based search engine on σ
$B_\sigma \leftarrow R_\sigma$
for each page $p \in R_\sigma$ **do**
  $\Gamma^+(p) \leftarrow$ the set of all pages p points to
  $\Gamma^-(p) \leftarrow$ the set of all pages pointing to p
  $B_\sigma \leftarrow B_\sigma \cup \Gamma^+(p)$
end for
if $|\Gamma^-(p)| \leq d$ **then**
  $B_\sigma \leftarrow B_\sigma \cup \Gamma^-(p)$
else
  $B_\sigma \leftarrow B_\sigma \cup$ an arbitrary set of d pages from $\Gamma^-(p)$
end if
return B_σ

For each page in the base page set B_σ, it is associated with two non-negative scores, *authority score* and *hub score*. The initial values for all scores are 1. Kleinberg uses a formula to update these two scores until they are stable:

$$\begin{aligned} authority\_score(p) &\leftarrow \sum_{(q,p)\in E} hub\_score(q) \\ hub\_score(p) &\leftarrow \sum_{(p,q)\in E} authority\_score(q) \end{aligned}$$

That is, authority scores will be transferred along edges to update corresponding hub scores while hub scores are transferred along edges reversely to update corresponding authority score. Each run finishes, these two scores are normalized to

$\sum_{p\in B_\sigma}(authority\_score(p))^2 = 1$, $\sum_{p\in B_\sigma}(hub\_score(p))^2 = 1$. The larger the scores are, the "better" the authorities and hub are respectively.

Kleinberg also proved that the authority and hub scores can always converge if they are initially assigned to 1.

3.3.2 Vincent D. Blondel, *et al.* , 2004

Based on the intuition of two vertices in different graphs are "similar" if the vertices in their neighborhoods are "similar". Blondel *et al.* [3] introduced a similarity measure of vertices between directed graphs, which is a generalization of Kleinberg's HITS algorithm.

Similar as Kleinberg's authority and hub scores updating, Blondel proposed his similarity updating equations as Equation 3.1 [3]:

$$sim_{ij} \leftarrow \sum_{(r,i)\in E_B,(s,j)\in E_A} sim_{rs} + \sum_{(i,r)\in E_B,(j,s)\in E_A} sim_{rs} \tag{3.1}$$

That is, if vertex i and j are pointing to r and s respectively, the similarity between r and s will be taken into account for the similarity between i and j at the next round. The similar situation happens for the vertices pointed by i and j.

Two directed graphs G_A and G_B can be presented by adjacency matrices A and B respectively. The number of vertices of G_A is n_A, and n_B for G_B. Blondel defines an $n_A \times n_B$ matrix S to represent the similarity of every vertex pair between G_A and G_B. Then the previous equation can be written into a compact matrix form as shown in Equation 3.2.

$$X \leftarrow BXA^T + B^TXA, \tag{3.2}$$

where all entries of the similarity matrix X are initialized to 1. A stable similarity matrix can be obtained by a limited number of iterations of updating the equation. Extensions of this measurement have been proposed with emphasis in different aspects.

More precisely, Blondel's algorithm for measuring the similarity between graph vertices is shown in Algorithm 2:

In Algorithm 2, the matrix norm $||\cdot||_F$ is the Euclidean or Frobenius norm which is equal to the square root of the sum of all squared entries. Additionally, Blondel proved that the subsequences X_{2k} and X_{2k+1} converge to Z_{even} and Z_{odd}.

Figure 3.2 shows an example of two graphs G_A, G_B [3]. And their similarity matrix is shown in Equation 3.3 [3].

Algorithm 2 GET_SIMILARITY_MATRICES_OF_GRAPHS

$X_0 \leftarrow \mathbf{1}$
Iterate an even number of times

$$X_{k+1} = \frac{BX_kA^T + B^TZ_kA}{||BX_kA^T + B^TZ_kA||_F}$$

and stop upon convergence
return the last value of X_k

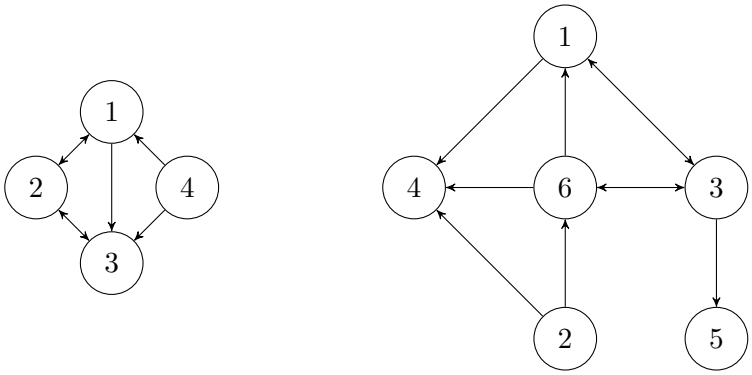


Figure 3.2: Two graphs G_A and G_B for vertices similarity computation [3]

$$\begin{bmatrix} 0.2636 & 0.2786 & 0.2723 & 0.1289 \\ 0.1286 & 0.1286 & 0.0624 & 0.1268 \\ 0.2904 & 0.3115 & 0.2825 & 0.1667 \\ 0.1540 & 0.1701 & 0.2462 & 0 \\ 0.0634 & 0.0759 & 0.1018 & 0 \\ 0.3038 & 0.3011 & 0.2532 & 0.1999 \end{bmatrix} \tag{3.3}$$

Blondel connects his updating formula with classical power method to compute the principle eigenvector of a matrix:

$$vec(BXA^T) = (A \otimes B)vec(X),$$

where *vec* is the matrix-to-vector operator which transforms a matrix into a vector by taking its columns one by one, and $\otimes$ is the Kronecker product:

$$vec(A) = [a_{11}, ..., a_{m1}, a_{12}, ..., a_{m2}, ..., a_{1n}, ..., a_{mn}]^T$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix} \tag{3.4}$$

Therefore, the updating formula can be rewritten as

$$vec(X) \leftarrow (A \otimes B + A^T \otimes B^T) vec(X).$$

If we use $x = vec(X)$ and $M = (A \otimes B + A^T \otimes B^T)$, the graph vertices similarity updating formula is shown in Equation 3.5 if normalization also applies.

$$x \leftarrow \frac{Mx}{||Mx||_F} \tag{3.5}$$

Thus, Equation 3.5 is a matrix analogue to how to compute the eigenvector of the Matrix M by using classical power method [16]. Therefore, if $x_0 = \mathbf{1}$ which is not orthogonal to the principal eigenvector of M (otherwise the principal eigenvector is $\mathbf{0}$), $\frac{Mx}{||Mx||_F}$ converges to the principal eigenvector [16].

3.3.3 M. Heymans and A.K. Singh, 2003

From the vertex's point of view, in addition to the similarity of the *connected* neighborhoods, Heymans *et al.* [15] considered positive flows from the *not connected* neighborhood and penalty flows from the "mismatched" neighborhood as well. These additional positive flows and penalties can be directly added on the Equation 3.1, thus Heymans's updating formula

is obtained as shown in Equation 3.6

$$\begin{aligned} sim_{ij} \leftarrow & \sum_{(r,i)\in E_B,(s,j)\in E_A} sim_{rs} + \sum_{(i,r)\in E_B,(j,s)\in E_A} sim_{rs} \\ & + \sum_{(r,i)\notin E_B,(s,j)\notin E_A} sim_{rs} + \sum_{(i,r)\notin E_B,(j,s)\notin E_A} sim_{rs} \\ & - \sum_{(r,i)\in E_B,(s,j)\notin E_A} sim_{rs} + \sum_{(i,r)\in E_B,(j,s)\notin E_A} sim_{rs} \\ & - \sum_{(r,i)\notin E_B,(s,j)\in E_A} sim_{rs} + \sum_{(i,r)\notin E_B,(j,s)\in E_A} sim_{rs} \end{aligned} \tag{3.6}$$

By using matrix form, the updating equation can be rewritten into a more compact form. Let I_B denote the matrix of all ones with the same dimension as B. The similarity matrix X is computed as [35]:

$$X \leftarrow BXA^T + B^TXA \tag{3.7}$$
$$+ (I_B - B)X(I_A - A)^T + (I_B - B)^TX(I_A - A) \tag{3.8}$$
$$- BX(I_A - A)^T - B^TX(I_A - A) \tag{3.9}$$
$$- (I_B - B)XA^T - (I_B - B)^TXA \tag{3.10}$$

Equation 3.7 adds the scores from the *connected* neighbors, which is the same as Equation 3.2 in [3]; Equation 3.8 adds the scores from the *not connected* neighbors; Equation 3.9 subtracts the score from the *mismatched* neighbors such that vertex r is mismatched to s if r is the neighbor of i but s isn't such a case for j; Equation 3.10 subtracts the score from the *mismatched* neighbors similar to Equation 3.9 [35].

This complement updating method for vertices similarity also converges. When applying vectorization to the similarity matrix X, the updating equation is [35]:

$$\begin{aligned} vec(X) \leftarrow & (A\otimes B + A^T\otimes B^T + (I_A - A)\otimes(I_B - B) + (I_A - A)^T\otimes(I_B - B^T) \\ & -(I_A - A)\otimes B - (I_A - A)^T\otimes B^T - A\otimes(I_B - B) - A^T\otimes(I_B - B)^T)vec(X). \end{aligned}$$

3.3.4 L.A. Zager and G.C. Verghese, 2008

From the edge's point of view, Zager *et al.* [36] brought forward a "vertex-edge" score. Before this work, all of the iterative methods to calculate the similarity between graph

vertices only include *node similarity*. Zager is the first one who consider the similarity of edges in iterative graph similarity framework.

Similar as nodes, two edges are considered to be "similar" if their starting and ending vertices are "similar", respectively. Let y_{ij} denote the similarity between edge i in Graph A and j in B, and let $s(i)$ and $e(i)$ denote the starting and ending vertices of edge i respectively. The updating equation for similarities of vertices and edges take the following form as shown in Equation 3.11 [36].

$$\begin{aligned} y_{ij} &\leftarrow x_{s(i)s(j)} + x_{e(i)e(j)} \\ x_{ij} &\leftarrow \sum_{e(k)=i,e(l)=j} y_{kl} + \sum_{s(k)=i,s(l)=j} y_{kl} \end{aligned} \tag{3.11}$$

In [35], Zager also gives the matrix form of the linear iterative updating framework between vertex similarity and edge similarity:

$$\begin{aligned} Y &\leftarrow B_S^T X A_S + B_T^T X A_T \\ X &\leftarrow B_S Y A_S^T + B_T Y A_T^T , \end{aligned}$$

where X is still the vertex similarity matrix, Y is the edge similarity matrix, A_S is the edge-source matrix of graph A, and A_T is the edge-terminus matrix of graph A. If expanding the coupled X and Y updating formulations, X turns out to be irrelative to edges [35]:

$$\begin{aligned} X &\leftarrow BXA^T + B^T XA \\ &+ D_{B_S} X D_{A_S} + D_{B_T} X D_{A_T} , \end{aligned}$$

where D_{B_S}/D_{B_T} is the diagonal matrix with the i-th diagonal entry equal to vertex i's out-degree/in-degree.

3.3.5 Summary of Existing Iterative Methods on Subgraph Matching

Many applications use graph matching. For example, Blondel *et al.* [3] used a similarity score to extract synonyms automatically. Le Saux *et al.* [28] proposed a graph matching based classifier for image processing.

Unluckily, these existing Iterative methods on subgraph matching could only be applied on directed graphs, and they also assign very high similarity scores to vertices of high degree [7]. If we applied Blondel's method [3] on undirected graph, the similarity matrix X will converges to a matrix of rank 1 which violates the truth.

The idea of graph similarity in this thesis is inspired by [19, 3, 15]. In Chapter 5, We develop a fuzz Jaccard approach for subgraph matching problems, which not only outperforms existing methods, but also is applicable to undirected graph. Later, we tackle the computation problem, and apply graph matching on social network mapping.

Chapter 4

Profile Matching

We discuss the profile matching problem in this chapter. To measure the similarity between an email contact's profile and a social network account's profile, we can calculate the similarity based on the names and the email addresses of the two profiles. As names and email addresses are typically text strings, this can be achieved by adopting some existing similarity measures on strings.

4.1 The Two String Similarity Measures

In our study, user names and email addresses are often short strings. There are two kinds of similarity that we need to capture: the similarity of two strings without considering the possible substring relation, and a string is similar to a substring of the other string. We consider two popularly used measures, namely the Jaro-Winkler similarity, and the overlap similarity to address those two situations, respectively.

4.1.1 Jaro-Winkler String Similarity

The Jaro-Winkler similarity [33] is a similarity measure good for short strings. It is widely used in record linkage and duplicate detection.

For a string s, denote by $s[i]$ $(i > 0)$ the i-th character in s. Consider two strings s_1 and s_2. Two characters $s_1[i]$ and $s_2[j]$ are regarded **matched** if $s_1[i] = s_2[j]$ and $|i - j| \leq \left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$. Let m be the number of matching characters between s_1 and s_2. Let s_1' (s_2') be the list of matched characters in s_1 (s_2) in the sequence order of s_1 (s_2). Apparently,

s'_1 and s'_2 have the same length m. The **number of transpositions** t is the number of positions i such that $s'_1[i] \neq s'_2[i]$ $(1 \leq i \leq m)$ divided by 2, rounding down. Then, the **Jaro distance** [18] is defined as

$$JaroScore(s_1, s_2) = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right).$$

The Jaro-Winkler similarity is a variant of the Jaro distance, which favors strings sharing a common prefix. Specifically, the **Jaro-Winkler similarity** is defined as

$$\begin{aligned} JWScore(s_1, s_2) = \quad & JaroScore(s_1, s_2) \\ & +(l \cdot p \cdot (1 - JaroScore(s_1, s_2))), \end{aligned}$$

where l is the length of the common prefix between s_1 and s_2, up to a maximum of 4, and p is a **scaling factor** that determines the amount of adjustment towards the common prefixes. Typically, p is set to 0.1.

It is easy to see that the Jaro-Winkler similarity is normalized. A similarity value of 0 means no similarity at all, and a value of 1 means an exact match.

Example 3 (Jaro-Winkler similarity) *Consider two strings "martha" and "marhta". We have*

$$JaroScore(\texttt{martha}, \texttt{marhta}) = \frac{1}{3} \times \left(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}\right) = 0.944\,,$$

and

$$JWScore(\texttt{martha}, \texttt{marhta}) = 0.944 + (3 \times 0.1 \times (1 - 0.944)) = 0.961 \qquad \blacksquare$$

We consider the Jaro-Winkler similarity because it has been shown effective in detecting duplicate or almost duplicate names in record linkage. We also empirically test some other similarity measures for this purpose, such as the string version of the Dice similarity [10]. Their effectiveness on the real data sets are close to but weaker than that of Jaro-Winkler similarity.

4.1.2 Overlap String Similarity

The overlap similarity [22] is a commonly used string similarity measure. It returns a high score when one string is the substring of the other one.

In email systems and social networks, people sometimes only register their first names or last names as usernames. In such kind of case, overlap similarity can capture the "subname" feature.

Technically, given two strings s_1 and s_2, the **overlap similarity** is defined as

$$OvlpScore(s_1, s_2) = \frac{|bigram(s_1) \cap bigram(s_2)|}{\min(|bigram(s_1)|, |bigram(s_2)|)},$$

where for a string s, $bigram(s) = \{s[i]s[i+1] | 1 \leq i < |s|\}$ is the set of bigrams [20] in s.

Example 4 (Overlap similarity) *Consider strings "marh" and "marhta". It can be verified that*

$$\begin{aligned} OvlpScore(\texttt{marh}, \texttt{marhta}) &= \frac{|\{\texttt{ma}, \texttt{ar}, \texttt{rt}, \texttt{th}\} \cap \{\texttt{ma}, \texttt{ar}, \texttt{rh}, \texttt{ht}, \texttt{ta}\}|}{min(|\{\texttt{ma}, \texttt{ar}, \texttt{rt}, \texttt{th}\}|, |\{\texttt{ma}, \texttt{ar}, \texttt{rh}, \texttt{ht}, \texttt{ta}\}|)} \\ &= \frac{4}{min(4, 6)} = 1.00 \end{aligned}$$ ■

The overlap similarity ranges from 0 to 1. The value is 0 if two strings are not similar at all; while the value is 1 if either two strings are identical or one string is a substring of the other one. We consider the overlap similarity because it is capable of capturing the similarity between a string and its substrings.

4.2 Integrating the Two String Similarity Measures

The two similarity measures have different strengths. In our method, we integrate them by affine combination to achieve a profile similarity score. Please note that both similarity measures are in the range of 0 to 1, and the larger the similarity value, the more similar two strings are.

We define the **profile similarity** between two strings s_1 and s_2, which can both be email addresses, display names, or any other corresponding attributes in email and social network profiles, as

$$\begin{aligned} ProfSim(s_1, s_2) = \; & \alpha \cdot JWScore(s_1, s_2) \\ & +(1-\alpha) \cdot OvlpScore(s_1, s_2), \end{aligned} \tag{4.1}$$

where $0 \leq \alpha \leq 1$. We learn the parameter value for α empirically using a set of training data, as described in Chapter 6. In the experiment, we iterate α from a range of 0 to 1 with a step of 0.1.

Chapter 5

Graph Matching

Profile matching can identify email correspondents in a social network only if the correspondents provide the same or very similar information in both the email and social network profiles. If a correspondent uses a different email address and/or name in the profiles, profile matching may not work.

In this section, we explore the graph matching approach to identifying email correspondents in a social network. We first present a universal connection heuristic. Then, we formulate the graph matching problem, present our approach with the proof of the convergence of our method, and integrate our graph matching approach with the profile matching approach.

5.1 The Universal Connection Heuristic

Heuristically, if two persons communicate well by email, likely they may be connected in a social network, and vice versa. We call this the **universal connection heuristic**.

Example 5 (The universal connection heuristic) *Consider our motivation example in Figure 1.1 again. Suppose Doe and Cathy in the email network are matched with Doe and Kathy in the social network by profile matching.*

By searching the neighbors in the social network (Figure 1.1(b)), we can find out that Doe and Kathy have a common neighbor "Super gamer". Interestingly, Doe and Cathy in the email network (Figure 1.1(a)) also have a common neighbor, "William". Heuristically, "William" in the email network and "Super gamer" in the social network may belong to the

same person. In other words, we may map email correspondent "William" to social network account "Super gamer". ■

According to the universal connection heuristic, comparing the email network and the social network may provide us some hints in finding email correspondents in the social network.

5.2 The Graph Matching Problem

Using all emails sent by and to a given user u, we can obtain the set of u's contacts. Moreover, by analyzing the sent-to and cc fields of the emails, we can build connections between u's contacts. The u's **email contact graph** is a graph $G_u = (C_u, E_u)$, where C_u is the set of u's contacts, and $(v_1, v_2) \in E_u$ if (1) $v_1 = u$ or $v_2 = u$; or (2) there is an email sent by or to u where both v_1 and v_2 are recipients. Here, an email address is a recipient if the address is listed in either the `sent-to` field or the `cc` field.

For email address u and all of u's contacts, we search the social network G_N using their profiles, and obtain a set of social network accounts V_{C_u} that are possible matches. By visiting the pages of those accounts, we can also know their friends in the social network. Let $V_u = V_{C_u} \cup \{v | v \text{ is a friend of } w, w \in V_{C_u}\}$ be the set of accounts in the social network that are either possible matches of u and u's contacts, or their friends in the social network. We can construct a **social network subgraph** $SN_u = (V_u, E_u)$ on V_u such that for $v_1, v_2 \in V_u$, $(v_1, v_2) \in E_u$ if v_1 and v_2 are friends in the social network, i.e., $(v_1, v_2) \in G_N$, and at least one of v_1 and v_2 is in V_{C_u}.

Now, the problem of finding email correspondents using graph matching is to find, for each email address $v \in C_u$, the social network accounts in V_u that most likely belong to the owner of v.

One may think that we can find some isomorphic subgraphs between G_u and SN_u to solve the problem. However, this is infeasible in practice, since it is not necessary that two persons exchanging emails are also connected in the social network, or vice versa. A method based on isomorphic subgraphs assumes a too strong assumption of the completeness and consistency of the information in the email contact graph and the social network.

Under the universal connection heuristic, a vertex v_1 in G_u and a vertex v_2 in SN_u are similar if many neighbors of v_1 in G_u can find similar peers in the neighbors of v_2 in SN_u, and vice versa. We formulate this idea as follows.

Let S be a $|C_u| \times |V_u|$ matrix such that $s_{v,w}$ is the **graph matching similarity** between vertex $v \in C_u$ and $w \in V_u$. Initially, we set

$$S^{(0)} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{5.1}$$

Let A and B be the adjacency matrices of graphs G_u and SN_u, respectively. Let f_{iter} be a function that refines the graph matching similarity. Then, we iteratively evaluate S by

$$S^{(i+1)} = f_{iter}(A, B, S^{(i)}) \tag{5.2}$$

The iteration continues until the graph matching similarity matrix converges.

The similar framework of evaluating a similarity matrix has been used in may previous studies. For example, Blondel *et al.* [3] used $S^{(i+1)} = AS^{(i)}B$. This method is simple. However, it assigns very high similarity scores to vertices of high degree [7]. In addition, the method can only be applied to directed graphs. In the case of undirected graphs, the similarity matrix S converges to be a matrix of rank 1. That is, $S^{+\infty} = I \cdot J^T$ where I and J are both column vectors. Clearly, it cannot be used to solve our problem. Next, we will develop a function f_{iter} for our problem.

5.3 Computing Graph Matching Similarity

Fuzzy Jaccard similarity [32] can measure the similarity between two disjoint sets of vertices in a graph. For two disjoint sets of vertices T_1 and T_2 in a graph, the **fuzzy Jaccard similarity** is defined as

$$FJ(T_1, T_2) = \frac{FI_{T_1,T_2}}{|T_1| + |T_2| - FI_{T_1,T_2}} \tag{5.3}$$

Here, *Fuzzy Intersection* (FI) is a value determined by the maximum weighted bipartite matching between T_1 and T_2. Let m be any bipartite matching and define $m(x, y) = 1$ if and only if (x, y) is covered by m or $m(x, y) = 0$ otherwise. The fuzzy intersection can thus be computed as

$$FI_{T_1,T_2} = \sum_{x \in T_1, y \in T_2} weight(x, y) M(x, y),$$

where $weight(x, y)$ is the weight of edge (x, y) and M is the maximum weighted bipartite matching, or

$$M = \arg\max_{m} \sum_{x \in T_1, y \in T_2} weight(x, y) m(x, y)$$

Let $weight(x, y)$ be the similarity score between vertex x and y, that is, $weight(x, y) = s_{x,y}$. Since $s_{x,y} \in [0, 1]$, we have $0 \leq FI_{T_1,T_2} \leq min(|T_1|, |T_2|)$.

We can rewrite Equation 5.3 as

$$FJ(T_1, T_2) = \frac{\sum_{x \in T_1, y \in T_2} s_{x,y} M(x, y)}{|T_1| + |T_2| - \sum_{x \in T_1, y \in T_2} s_{x,y} M(x, y)} \tag{5.4}$$

The range of $FJ(T_1, T_2)$ is $[0, 1]$.

Heuristically, two vertices are similar if their neighbors are similar. Therefore, an intuitive solution to measure the similarity of two vertices is to compute the fuzzy Jaccard similarity of their neighbors. Formally, let $N(v)$ and $N(w)$ be the sets of neighbors of vertices v and w respectively. Using Equation 5.4, we can define the similarity of vertices $s_{v,w}$ as

$$\begin{aligned} s_{v,w} &= FJ(N(v), N(w)) \\ &= \frac{\sum_{x \in N(v), y \in N(w)} s_{x,y} M(x, y)}{|N(v)| + |N(w)| - \sum_{x \in N(v), y \in N(w)} s_{x,y} M(x, y)} \end{aligned} \tag{5.5}$$

In each iteration, we apply Equation 5.5 to update each entry of the similarity matrix. We rewrite Equation 5.5 into an iterative form.

$$\begin{aligned} s^{i+1}_{v,w} &= FJ^i(N(v), N(w)) \\ &= \frac{\sum_{x \in N(v), y \in N(w)} s^i_{x,y} M^i_{v,w}(x, y)}{|N(v)| + |N(w)| - \sum_{x \in N(v), y \in N(w)} s^i_{x,y} M^i_{v,w}(x, y)}, \end{aligned} \tag{5.6}$$

where $M^i_{v,w}(\cdot)$ denotes the maximum weighted bipartite matching between the neighbor sets of vertices v and w at the i^{th} iteration. $M^i_{v,w}(x, y) = 1$ if edge (x, y) is in the maximum matching, and 0 otherwise. We prove that our iterative method converges.

Theorem 1 (Convergence) *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. For each vertex pair (v, w), $v \in V_1$, $w \in V_2$, the sequence $\{s^i_{v,w}\}$ $(i = 1, 2, \ldots)$ converges.*

PROOF *We show by induction that the sequence* $\{s^i_{v,w}\}$ *is non-increasing.*

Basis. *According to Equation 5.1,* $s^0_{v,w} = 1$. *In the* 1^{st} *iteration, the value of maximum weighted bipartite matching for pair* (v, w) *equals the smaller size of neighbor sets of* v *and* w, *since the initial similarity of every two vertices is* 1. *Thus, we have* $s^1_{v,w} = \frac{min(|N(v)|,|N(w)|)}{max(|N(v)|,|N(w)|)} \leq 1 = s^0_{v,w}$.

Inductive step. *Assume that the sequence* $(s^i_{v,w})^k_{i=0}$ *is non-increasing. In other words,* $s^k_{x,y} \leq s^{k-1}_{x,y}$ *for any pair* (x, y). *Then, we have*

$$\sum_{x\in N(v),y\in N(w)} s^k_{x,y} M^k_{v,w}(x, y) \leq \sum_{x\in N(v),y\in N(w)} s^{k-1}_{x,y} M^k_{v,w}(x, y) \tag{5.7}$$

Since $M^{k-1}_{v,w}(\cdot)$ *is the maximum weighted matching in the* $(k-1)^{th}$ *iteration, for any matching* $m_{v,w}(\cdot)$,

$$\sum_{x\in N(v),y\in N(w)} s^{k-1}_{x,y} m_{v,w}(x, y) \leq \sum_{x\in N(v),y\in N(w)} s^{k-1}_{x,y} M^{k-1}_{v,w}(x, y)$$

Consequently, we have

$$\sum_{x\in N(v),y\in N(w)} s^{k-1}_{x,y} M^k_{v,w}(x, y) \leq \sum_{x\in N(v),y\in N(w)} s^{k-1}_{x,y} M^{k-1}_{v,w}(x, y) \tag{5.8}$$

Combining Equations 5.7 and 5.8, we have

$$\begin{aligned} s^{k+1}_{v,w} &= \sum_{x\in N(v),y\in N(w)} s^k_{x,y} M^k_{v,w}(x, y) \\ &\leq \sum_{x\in N(v),y\in N(w)} s^{k-1}_{x,y} M^{k-1}_{v,w}(x, y) = s^k_{v,w}. \end{aligned}$$

Apparently, $s^i_{v,w} \geq 0$ *for any* $i > 0$. *Therefore, the non-increasing sequence* $\{s^i_{v,w}\}$ *has a lower bound* 0. *Thus, the sequence converges.* ∎

Theorem 1 shows the correctness of our graph matching method. For the sake of efficiency, we conduct iterations until the changes in the graph matching similarity matrix are all smaller than a user-specified threshold $\epsilon > 0$ in the last iteration.

5.4 Enhancing Graph Matching Similarity

Equation 5.6 only considers the neighbors. To be more accurate, we borrow a similar idea from Heymans *et al.* [15], where the remaining vertices other than the neighbors are also

taken into account in the similarity computation. More generally, we consider a directed graph G where each vertex $j \in G$ has an in-neighbors set $N(j, \text{in})$ and an out-neighbors set $N(j, \text{out})$. Let $\tilde{N}(j, \text{in})$ and $\tilde{N}(j, \text{out})$ be the complement set of $N(j, \text{in})$ and $N(j, \text{out})$, respectively. The enhanced vertices similarity can thus be computed by the following equation

$$\begin{aligned} \dot{s}_{v,w}^{i+1} \;=\; \frac{1}{4} \;& (FJ^i(N(v,\text{in}), N(w,\text{in})) + FJ^i(N(v,\text{out}), N(w,\text{out})) \\ & + FJ^i(\tilde{N}(v,\text{in}), \tilde{N}(w,\text{in})) + FJ^i(\tilde{N}(v,\text{out}), \tilde{N}(w,\text{out})) \\ & - FJ^i(\tilde{N}(v,\text{in}), N(w,\text{out})) - FJ^i(N(v,\text{in}), \tilde{N}(w,\text{out})) \\ & - FJ^i(\tilde{N}(v,\text{out}), N(w,\text{in})) - FJ^i(N(v,\text{out}), \tilde{N}(w,\text{in}))) \end{aligned} \tag{5.9}$$

Since FJ has value from 0 to 1, our enhanced fuzzy Jaccard similarity has value from -1 to 1. In our graph matching similarity problem, the higher the similarity value between two vertices, the more similar they are.

For undirected graph, there is no difference between in-neighbors and out-neighbors for a given vertex. Formally, let $\tilde{N}(j)$ be the complement set of $N(j)$, that is,

$$\begin{aligned} \tilde{N}(j) \cup N(j) &= V \qquad \forall G(V,E),\ \forall j \in V \\ \tilde{N}(j) \cap N(j) &= \emptyset \qquad \forall G(V,E),\ \forall j \in V \end{aligned}$$

The enhanced vertices similarity (undirected version) can thus be computed by the following equation

$$\begin{aligned} \dot{s}_{v,w}^{i+1} \;=\; \frac{1}{2} \;& (FJ^i(N(v), N(w)) \\ & + FJ^i(\tilde{N}(v), \tilde{N}(w)) \\ & - FJ^i(\tilde{N}(v), N(w)) \\ & - FJ^i(N(v), \tilde{N}(w))) \end{aligned} \tag{5.10}$$

Here, common not-connections provide positive score as well as common connections (neighbors), but mismatched neighbors are penalizing the similarity score instead of "abstentions" in Equation 5.6. Since the range of $FJ(\cdot)$ is $[0, 1]$, our enhanced fuzzy Jaccard similarity (undirected version) has value from -1 to 1.

5.5 Integrating Profile Similarity and Graph Matching Similarity

The profile similarity and the graph matching similarity capture different characteristics of email and social network accounts. To take the advantages of both, we use the affine combination of the two similarity scores. Formally, for email account v and a social network account w, the overall similarity between them is calculated by

$$Sim^i(v, w) = \gamma ProfSim(v, w) + (1 - \gamma)s^i_{v,w}, \tag{5.11}$$

where $s^i_{v,w}$ is the graph matching similarity between v and w, and $0 \leq \gamma \leq 1$, and Sim is also a $|C_u| \times |V_u|$ matrix of the same size as S (in Section 5.2). As the postprocessing, we normalize the graph matching similarity matrix such that for each row in S, the sum of the squared similarity scores equals 1. We learn the parameter value for γ empirically using a set of training data, as discussed in Chapter 6.

Chapter 6

Experimental Results

In this section, we report an empirical study of our methods on two real data sets. We first describe how the real data sets were collected, and how the evaluation was conducted. Then, we report the effectiveness of profile matching and graph matching as well as their combinations.

6.1 Real Data Set Preparation

To test our methods, we need both email data and social network data about individuals. It is very difficult to obtain such data. Since there are not sufficient and well accepted statistics about such data, synthetic data generated based on some simple probabilistic models may not approach the reality, and thus may not be reliable.

Luckily, two volunteers agreed to let us test our methods on their email and Facebook data. Under the agreement, the two volunteers have to be kept anonymous all the time, and no details about their specific communication or friends can be disclosed. Only aggregate statistics can be reported in this thesis. These two volunteers know each other, but they only share less than 30 email contacts. In other words, they do not have a heavy overlap in their friends.

We call the two volunteers A and B hereafter. For each volunteer, we downloaded all her emails and constructed an email network by creating a vertex for any contact and linking two contacts by an edge if the two contacts are involved in an email. For each contact, we used both the email address and the display name (if available) as the profile information. The email contacts owned by A and B are also in the email networks. We also need to find

a match for them.

To construct the social network for each volunteer, we used the emails of the contacts to search Facebook. If no exact match was returned, we used the display names to search again, and obtained the candidate matches. We only used the Facebook accounts of such candidates, since the search results may contain Facebook pages that have no specific users behind. Moreover, some popular names may return hundreds or even thousands of Facebook accounts. In such cases, we selected the first 50 of them. We then connected all returned accounts if they have friendship relationship by visiting their personal Facebook pages. In the rest of this section, we call the social network formed as such the **Facebook network**. Note that the Facebook network graph may not be a connected graph.

The statistics of the two data sets, denoted by D_A and D_B, respectively, are summarized in Table 6.1. While the social networks contain the possible candidate matches returned from searching Facebook using the display names, the common vertices ($V_M \cap V_N$) and the common edges ($E_M \cap E_N$) were calculated according to the ground truth provided by the volunteers. The statistics show that the a non-trivial percentage of one's email correspondents appear in the Facebook networks (18.66% for A and 45.58% for B), but only a relatively smaller portion of email correspondents are also connected in the Facebook networks (1.33% for D_A and 15.73% for D_B). This observation also demonstrates the task of finding email correspondents in social networks is meaningful.

	Email network		Social network			
	$\|V_M\|$	$\|E_M\|$	$\|V_N\|$	$\|E_N\|$	$\|V_M \cap V_N\|$	$\|E_M \cap E_N\|$
D_A	2439	180524	7575	440325	455	2392
D_B	452	4676	11176	566491	206	736

Table 6.1: The statistics of the two real data sets.

6.2 Evaluation Method

We evaluated the effectiveness of profile matching, graph matching, and their combination. On each data set, we computed the similarity between an email contact in the email network and a Facebook account in the Facebook network using the matching method under test. For each email account that there is a corresponding Facebook account in the Facebook network, determined by the corresponding volunteer, we used the top-k Facebook accounts

in similarity as the **matching result**, where k is called the **answer set size**. If the top-k results contain the correct Facebook account, the matching is counted successful.

The **matching accuracy** is thus defined as the percentage of successful matchings. We do not consider the matching recall since the ground truth is incomplete. We report the matching accuracy with respect to various answer set size (k).

One challenge in evaluation is that even the volunteers do not know the complete ground truth. In other words, the volunteers do not know exactly whether their email contacts have Facebook accounts or not. The volunteers only can determine whether whether a Facebook account belongs to an email contact based on the Facebook personal page. Consequently, we cannot measure the recall of any methods.

All our experiments were conducted on an Apple Macbook Pro computer with a 2.4 GHz Intel Core 2 Duo CPU, 4 GB of 1066 MHz DDR3 SDRAM main memory, running the 64 bit Mac OS X v10.6 Snow Leopard operating system. Our programs were implemented in C++. By default, we set $\epsilon = 10^{-5}$ as the termination condition in our iterative method.

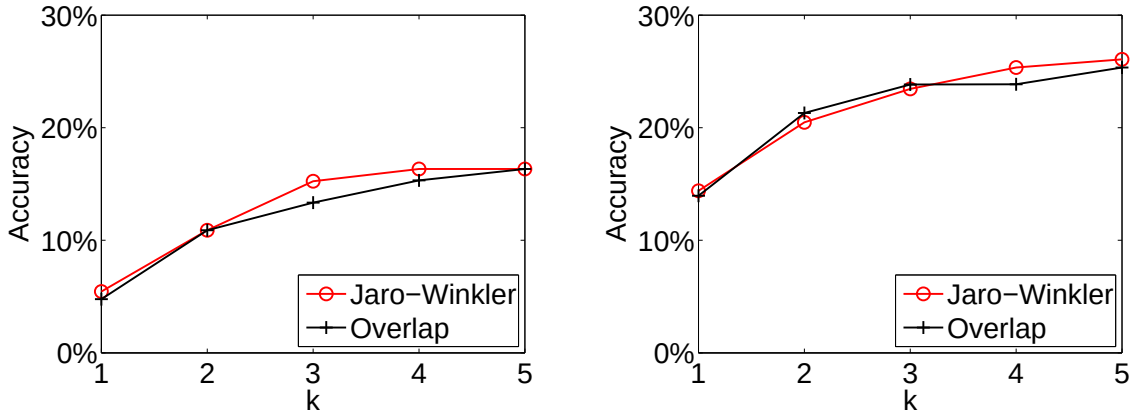
6.3 Effectiveness of Profile Matching

In this section, we report the evaluation results of profile matching. There are many string similarity measures that may be used for profile matching. This section by no means tries to evaluate all or the best string similarity measures for our problem. As discussed in Chapter 4, we use the Jaro-Winkler similarity and the overlap similarity to address the similarity between two independent strings and that between a string and its substrings, respectively. The evaluation here is to understand whether the two similarity measures complement each other, and how the two can be integrated to form a profile similarity measure.

Figure 6.1 shows the matching accuracy of the two similarity measures with respect to various answer set size on the two real data sets. On both data sets, the Jaro-Winkler similarity outperforms the overlap similarity when the answer set size ranges from 1 to 5. The accuracy of both methods are low, less than 30% on both data set even when the answer set size $k = 5$. This clearly illustrates that profile matching only is insufficient in tackling the problem of finding email correspondents in social networks.

To understand how much the two similarity measures complement each other, Figure 6.2 shows the Jaccard coefficient on the matching results of the two similarity measures. The

results indicate that the two similarity measures are in general correlated. This also suggests that the effectiveness of profile matching may not be very sensitive to the choice of specific string similarity measures.



(a) Dataset D_A (b) Dataset D_B

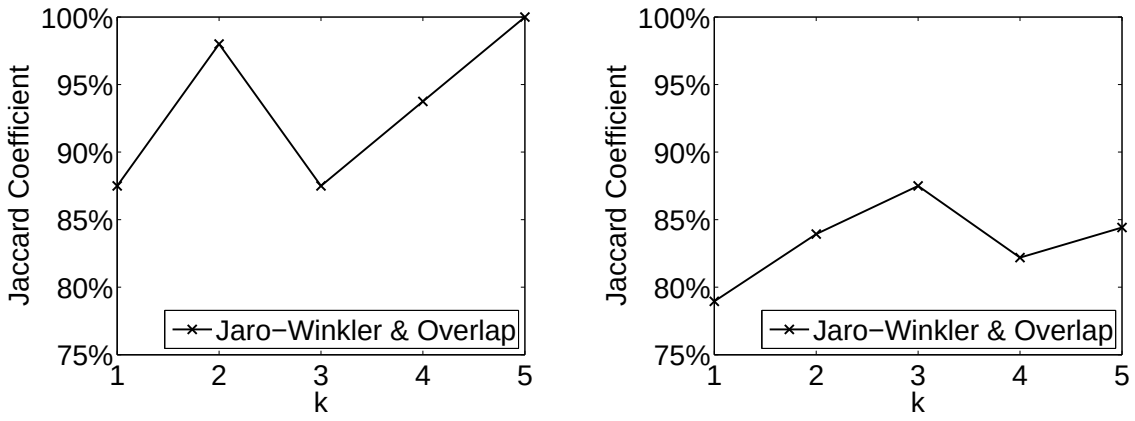
Figure 6.1: The matching accuracy of the two string similarity measures.

To learn the parameter α in the profile similarity (Equation 4.1), we compute the matching accuracy with respect to α, varying from 0 to 1 of step 0.1. Figure 6.3 shows the results. For answer set size in the range from 1 to 5, the matching accuracy is the highest when $\alpha = 0.8$. Therefore, we set $\alpha = 0.8$ by default.

6.4 Effectiveness of Graph Matching

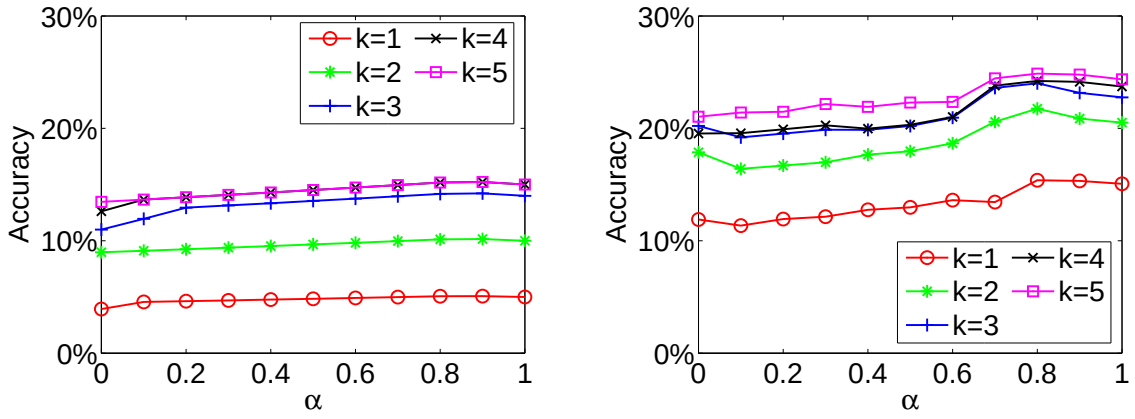
In this section, we report the evaluation results of graph matching. We compare the fuzzy Jaccard similarity (Section 5.3, denoted by FJ) and the state-of-the-art graph matching methods developed by Blondel *et al.* [3] (denoted by BV), Zager *et al.* [36] (denoted by ZV) and Heymans *et al.* [15] (denoted by HS). All methods were implemented in C++.

The synthetic graph of n vertices is generated by using the random graph model from Erdös and Rényi [13] where each pair of vertices is connected with a probability p, denoted as graph $G_{n,p}$. More precisely, each entry of $G_{n,p}$'s adjacent matrix has a value of 1 with probability p and 0 with probability $1 - p$. To construct a subgraph G_m induced from $G_{n,p}$,



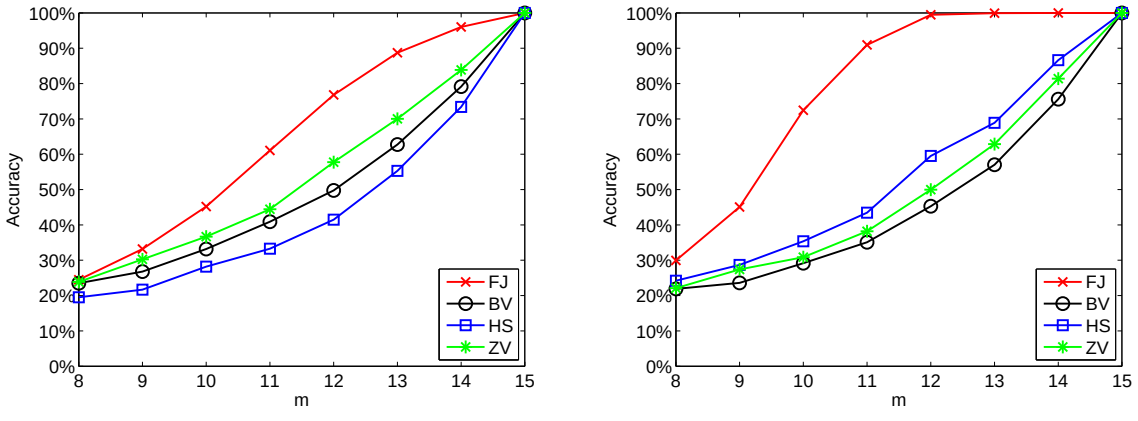
(a) Dataset D_A (b) Dataset D_B

Figure 6.2: The Jaccard coefficient between the two string similarity measures in profile matching.

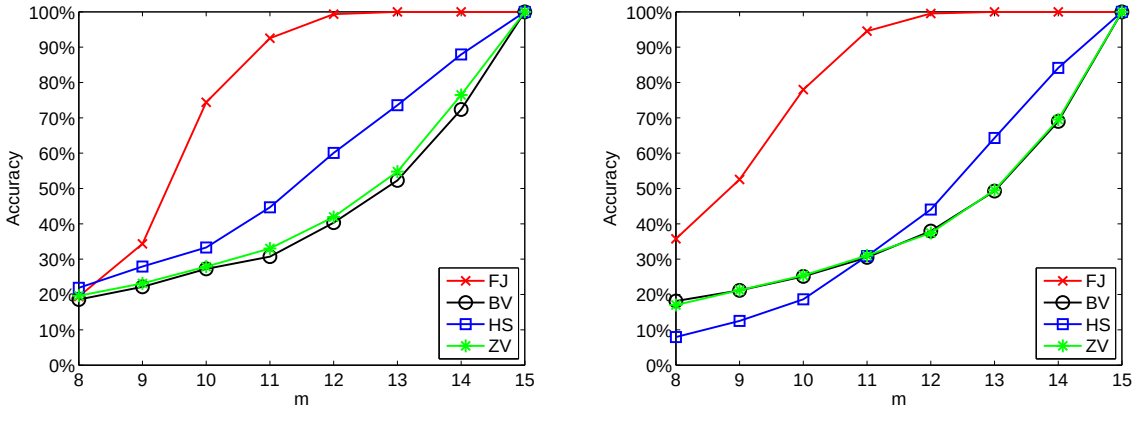


(a) Dataset D_A (b) Dataset D_B

Figure 6.3: Profile similarity parameter tuning.

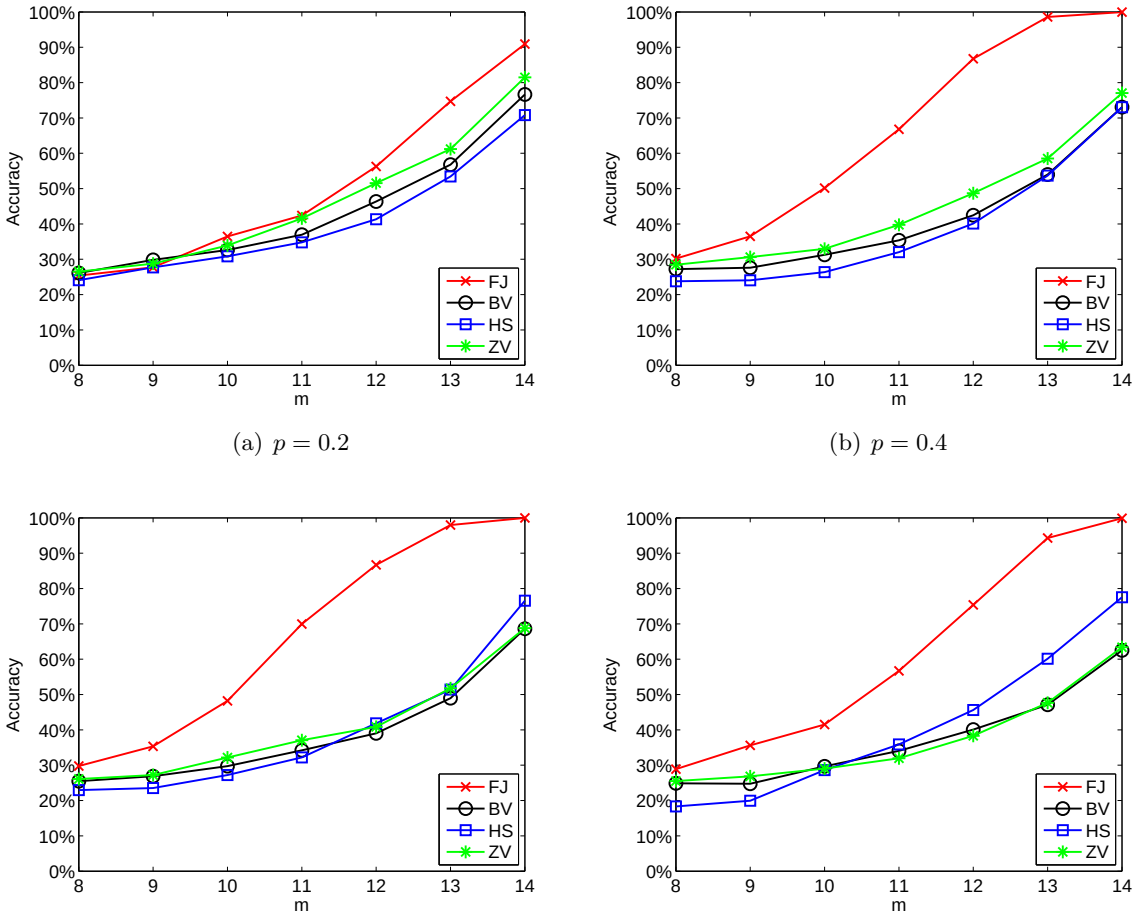


(a) $p = 0.2$ (b) $p = 0.4$



(c) $p = 0.6$ (d) $p = 0.8$

Figure 6.4: Accuracy comparison of four methods in the 1^{st} experiment



(a) $p = 0.2$

(b) $p = 0.4$

(c) $p = 0.6$

(d) $p = 0.8$

Figure 6.5: Accuracy comparison of four methods in the 2^{nd} experiment

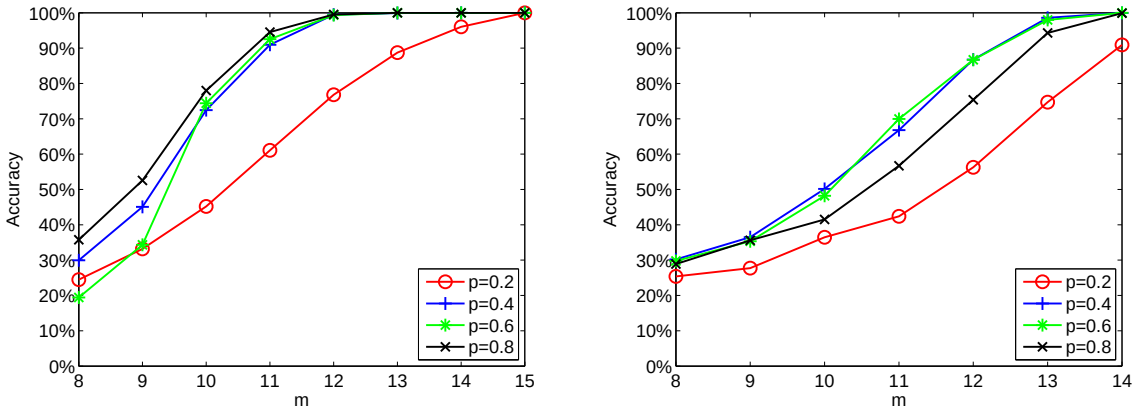
we randomly select a subset of vertices with size of m and maintain their connectivity in $G_{n,p}$. Note that G_m may not be a connected graph.

We conduct two graph matching experiments on $G_{n,p}$. The first one is to match one of its subgraph G_m with $G_{n,p}$ and the other one is to match its two random subgraphs G_{m_1} and G_{m_2} with the same size, say $m_1 = m_2$. The matching results are returned after comparing the similarity of each pair of vertices in the two graphs. As before, we compute the matching accuracy which is the number of the matched pairs over the size of the subgraph.

In the first experiment, n is set to be 15 and m scales from 8 to 15. The value of the connectivity probability p is selected from a range of 0.2 to 0.8 with a step of 0.2. For each setting of (m, p), we generate 500 random subgraphs and compute the average matching accuracy by using our FJ and the other three methods, as shown in Fig. 6.4. We see that our FJ provides consistently better performance than others with a considerable gap.

In the second experiment, $m_1(m_2)$ scales from 8 to 14 (we omit the size of 15 to avoid the case when the two subgraphs are equivalences of the original graph). p has the same value as that in the first experiment. Again, we compute the average matching accuracy and compare our FJ with others. Fig. 6.5 demonstrates the superiority of FJ.

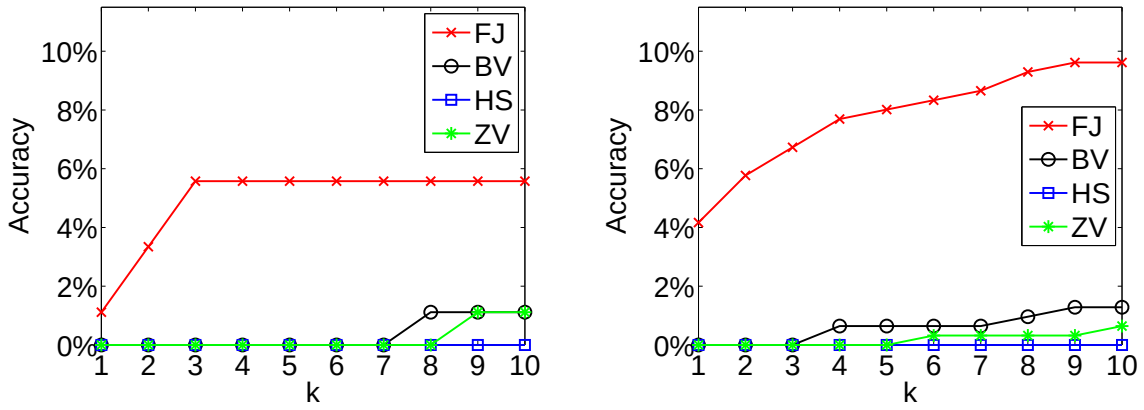
An important remark is that our FJ is robust to different subgraph sizes as that in other methods. This is particularly important as the practical applications, e.g., the social network have a huge number of users coexist in the network graph.



(a) The 1^{st} experiment (b) The 2^{nd} experiment

Figure 6.6: Summation of FJ's accuracy

We also apply different graph matching algorithms on real data sets. Figure 6.7 shows the results. The results clearly show that FJ outperforms the existing graph matching methods significantly with respect to a wide range of answer set size. In other words, our graph matching method is dramatically more effective the the other general graph matching methods on the problem of finding email correspondents in social networks.



(a) Dataset D_A (b) Dataset D_B

Figure 6.7: The matching accuracy of the graph matching methods.

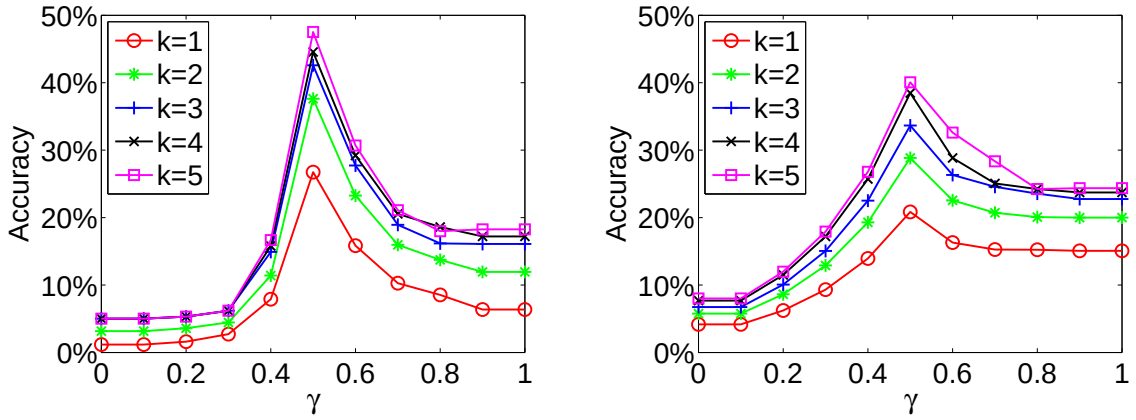
However, the matching accuracy is poor, less than 10%, for all graph matching methods tested. This is not surprising. There are many small subgraphs in the email network and the Facebook network that are similar to each other if only the subgraph structures are considered. This observation clearly shows that only graph matching is ineffective either to tackle the problem of finding email correspondents in social networks.

6.5 Effectiveness of Integrating Profile Matching and Graph Matching

In Section 5.5, we advocate integrating profile matching and graph matching. In this section, we report the evaluation results of this integrated approach.

First, we learn the value of parameter γ in Equation 5.11. Similar to learning the value of parameter α in profile matching, we compute the matching accuracy with respect to

γ, varying from 0 to 1 of step 0.1. Figure 6.8 shows the results. The results show that, empirically in our cases, the best performance is achieved by setting $\gamma = 0.5$.



(a) Dataset D_A (b) Dataset D_B

Figure 6.8: The matching accuracy of the integrated method with respect to parameter γ.

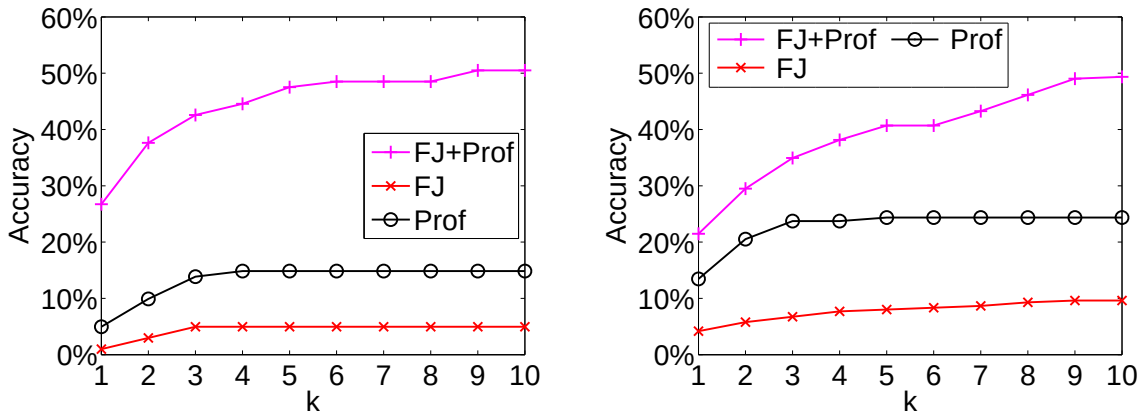
Figure 6.9 shows the matching accuracy of the integrated method. For the convenience of comparison, the figure also plots the accuracy of graph matching only and profile matching only. The results clearly show that the integrated method can achieve much better performance than graph matching only and profile matching only. The accuracy of the integrated method is even higher than the sum of the accuracies of graph matching only and profile matching only. The results verify that the integration of the two matching methods iteratively is effective.

6.6 The Effect of ϵ and the Runtime

As discussed at the end of Section 5.3, we use a parameter ϵ to control the termination of our iterative method. Figure 6.10 shows the accuracy on the two data sets with respect to ϵ. Please note that the ϵ is plotted in logarithmic scale.

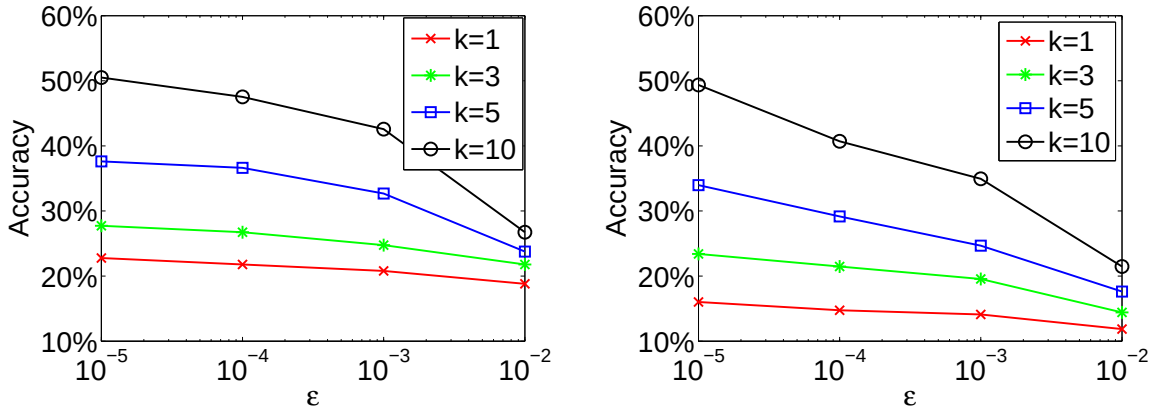
Clearly, the smaller the value of ϵ, the more accurate the results. However, the marginal improvement of lowering down ϵ decreases exponentially.

The runtime cost of our method is composed of two parts: the cost of crawling the social networks (Facebook in our experiments) and the cost of running the matching algorithms



(a) Dataset D_A (b) Dataset D_B

Figure 6.9: Comparison of different matching algorithms

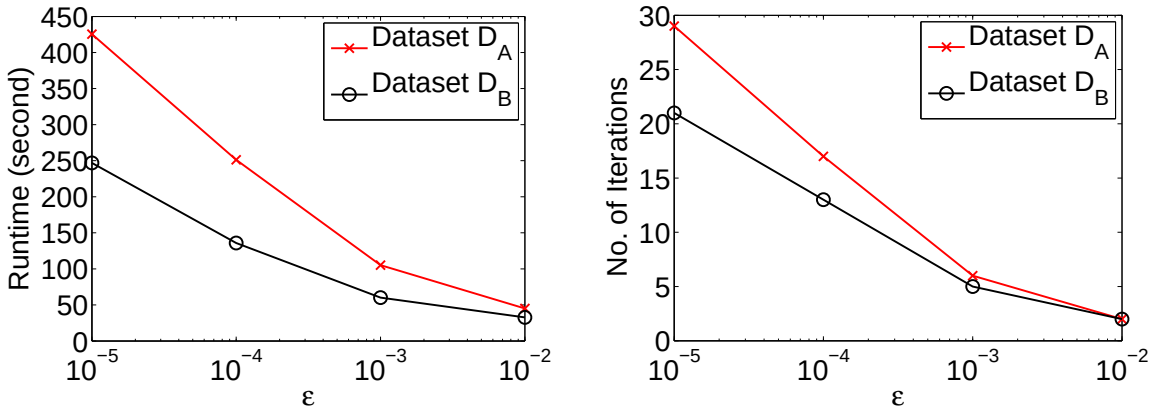


(a) Dataset D_A (b) Dataset D_B

Figure 6.10: Accuracy with respect to ϵ.

presented in this paper. The cost of crawling the social networks varies dramatically in different applications. For examples, on the one hand, if our method was run in Facebook, then the crawling time is largely ignorable. On the other hand, if one uses only one computer to crawl a social network that does not provide a proper API, it may take much time in crawling.

Since the crawling problem is orthogonal to the algorithms developed in this paper, we decided to focus on the runtime cost of the similarity computation. Thus, we report here the runtime that does not count the crawling cost.



(a) Runtime

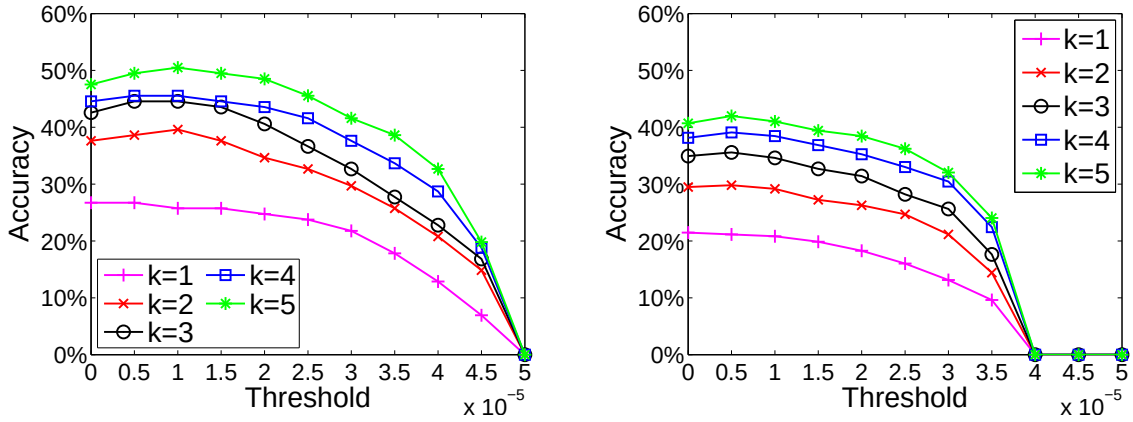
(b) Number of iterations

Figure 6.11: Runtime and number of iterations.

Figure 6.11 reports, with respect to ϵ, the runtime and the number of iterations of our method on the two data sets. Again, ϵ is plotted in the logarithmic scale. The results clearly show that parameter ϵ can be used to control the tradeoff between accuracy and efficiency. Please note that our method can be sped up substantially by distributed computing adopting similar techniques in many other matrix computation problems. Limited by space, we have to leave this as a future work.

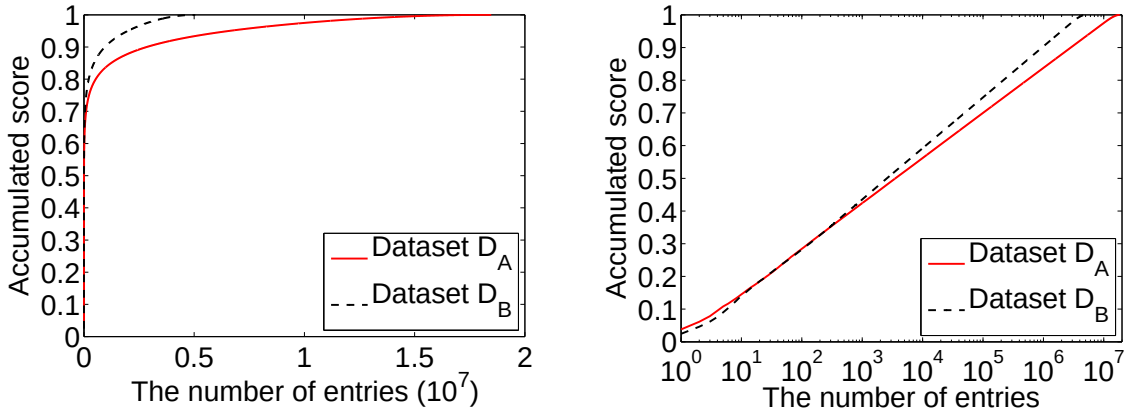
6.7 The Effect of the Threshold

As we make recommendations even if the similarity scores between an email contact and all of the social network accounts is low, those recommendations based on low similarity



(a) Dataset D_A (b) Dataset D_B

Figure 6.12: Accuracy with respect to the threshold.



(a) Dataset D_A (b) Dataset D_B

Figure 6.13: Similarity score distribution at convergence

scores may hurt the performance. Figure 6.12 shows the accuracy on the two data sets with respect to the threshold. The results show that the performance could be improved slightly if the threshold is set in the range of $[0.5, 1] * 10^{-5}$. However, if the threshold is set too high, it will eliminate more valid recommendations. One extreme situation is that the accuracy gets 0 if the threshold is higher than the maximal similarity score.

Figure 6.13 reports the accumulated similarity score when all similarity scores converge. As we discussed in Section 5.5, all of the similarity scores can be fetched in the matrix Sim, and we compute the matrix iteratively until it converges. The size of the similarity matrix of dataset D_A is 2439×7575, and that of dataset D_B is 452×11176. Therefore, there will be $18,475,425$ similarity scores for D_A, and $5,051,552$ similarity scores for D_B. We sort these scores in descending order, and accumulate them one by one. As we normalize the matrix at the postprocessing of every iteration, the final accumulated score is always 1. In Figure 6.13, we plot two figures, one with linear scale and another with logarithmic scale. From the results we obtained, the similarity score distribution is consistent with power-law distribution.

Chapter 7

Conclusions and Discussion

In this thesis, we tackled a practical and interesting problem of finding email correspondents in social networks. We considered two ways to tackle the problem. First, we considered using profile similarity that can be computed using string similarity. Second, we developed a novel graph matching similarity approach. Our experimental results showed that neither profile matching nor graph matching individually can solve the problem. Our integrated method can achieve much higher accuracy on the real data sets.

Although we only discussed the personal view version of the problem, our solution can be straightforwardly extended to tackle the general problem of finding email correspondents in social networks, where it is assumed that the whole email network and the social network are available. For the profile similarity computation, we can compare the profile of each email account with that of each social network account. For the graph matching similarity, we can use the two graphs to compute the graph matching similarity matrix. Besides the personal view and the whole network view, our solution can be applied in the cluster view as well. We can partition the email networks and social networks into several clusters according to users' education or workplace, then compute the similarity among users belonging to the same cluster.

In this thesis, we assumed that one person as only an account in a social network. In practice, this assumption may not hold for some social networks. One idea to break this assumption is to conduct "entity identification" in social networks, that is, identifying multiple accounts that are owned by the same person. This is an interesting problem for future study.

In our work, the vertices of email networks are connected because of their email communications. To simplify the problem, we only consider whether two vertices have an edge or not. One idea to improve the email networks is to assign weight on edges. Technically, we can assign two kinds of weight in our problem. One is the frequency of communication, that is how many emails interacted between two vertices, and the number of emails is assigned to weight of the edge of the two vertices. Another is the timestamps of emails. Heuristically, the latest emails have higher priority than the oldest emails, and the priority of emails affects the sender or receiver's priority. Therefore, when an email comes, it obtains a constant value which decreases by a given amount or by a given percentage daily. We can also use similar way to get weight of edges in social networks by referring messaging communication in social networks.

In contrast to weight of edges, it is also possible to assign direction to edges. Directions come naturally in email networks since every email always goes from its sender to its receiver. In social networks, directions cannot be retrieved directly, except Twitter in which every user has its followers and followings corresponding to incoming and outgoing vertices respectively. However, we still can fetch direction in social networks from users' communications. It is obvious to find that discussion happens everywhere in Facebook, for example discussion on one user's recently uploading photo, that on one's updated status, and so on. From such discussion, we could get some kind of following relationship among users and translate the following relationships into directions of edges.

Bibliography

[1] H. A. Almohamad and Salih O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):522–525, 1993.

[2] S. Appavu Alias Balamurugan, G. Athiappan, M. Muthu Pandian, and Ramasamy Rajaram. Classification methods in the detection of new suspicious emails. *Journal of Information and Knowledge Management(JIKM)*, 7(3):209–217, 2008.

[3] Vincent D. Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. In *SIAM Review (SIAM Rev.)*, pages 647–666, 2004.

[4] Horst Bunke. Graph matching: Theoretical foundations, algorithms, and applications. In *Version Interface 2000*, pages 82–88, 2000.

[5] Tibério S. Caetano, Julian John McAuley, Li Cheng, Quoc V. Le, and Alexander J. Smola. Learning graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):1048–1058, 2009.

[6] Vitor R. Carvalho and William W. Cohen. Preventing information leaks in email. In *SIAM Data Mining Conference (SDM)*, 2007.

[7] Thomas P. Cason, Pierre-Antoine Absil, and Paul Van Dooren. Review of similarity matrices and application to subgraph matching. In *Book of Abstracts of the 29th Benelux Meeting on Systems and Control*, page 109, 2010.

[8] William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78, 2003.

[9] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1367–1372, 2004.

[10] Lee R. Dice. Measure of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302, 1945.

[11] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19:248–264, April 1972.

[12] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19:1–16, 2007.

[13] P. Erdös and A. Rényi. On random graphs. *Publ. Math*, 6:290–297, 1959.

[14] Richard Wesley Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[15] M. Heymans and A.K. Singh. Deriving phylogenetic trees from the similairty analysis of metabolic pathways. *Bioinformatics*, 19(suppl 1):138–146, 2003.

[16] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge university press, 2005.

[17] P. Jaccard. *Nouvelles recherches sur la distribution florale*, volume 44. Bulletin de la Sociét*é* Vaudoise de la science naturelle, 1908.

[18] Matthew A. Jaro. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[19] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (J. ACM)*, 46(5):604–632, 1999.

[20] Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. Cognates can improve statistical translation models. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (HLT-NAACL)*, 2003.

[21] H.W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[22] Lawrence R. Lawlor. Overlap, Similarity, and Competition Coefficients. *Ecology*, 61(2):245–251, 1980.

[23] Matthew Michelson and Craig A. Knoblock. Semantic annotation of unstructured and ungrammatical text. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1091–1098, 2005.

[24] Chris Pal. Cc prediction with graphical models. In *3rd Conference on Email and Anti-Spam (CEAS)*, 2006.

[25] Jaccard Paul. *É*tude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[26] Maayan Roth, Assaf Ben-David, David Deutscher, Guy Flysher, Ilan Horn, Ari Leichtberg, Naty Leiser, Yossi Matias, and Ron Merom. Suggesting friends using the implicit social graph. In *16th ACM SIGKDD Conference on Knowlegde Discovery and Data Mining (KDD)*, pages 233–242, 2010.

[27] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approcah to filtering junk e-mail. In *Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Learning for Text Categorization*, 1998.

[28] Bertrand Le Saux and Horst Bunke. Feature selection for graph-based image classifiers. In *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA) (2)*, pages 147–154, 2005.

[29] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems (Inf. Syst.)*, 26(8):607–633, 2001.

[30] Julian R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (J. ACM)*, 23(1):31–42, 1976.

[31] Michaël A. van Wyk, Tariq S. Durrani, and Barend J. van Wyk. A rkhs interpolator-based graph matching algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):988–995, 2002.

[32] J. Wang, G. Li, and J. Feng. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE)*, 2011.

[33] William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, 1990.

[34] Shinjae Yoo, Yiming Yang, Frank Lin, and Il-Chul Moon. Mining social networks for personalized email prioritization. In *15th ACM SIGKDD Conference on Knowlegde Discovery and Data Mining (KDD)*, pages 967–976, 2009.

[35] Laura Zager. Graph Similarity and Matching. Master's thesis, Massachusetts Institute of Technology, USA, 2005.

[36] Laura A. Zager and George C. Verghese. Graph similarity scoring and matching. *Appl. Math. Lett.*, 21(1):86–94, 2008.