World Wide Web (2013) 16:195–218
DOI 10.1007/s11280-012-0168-2

Finding email correspondents in online social networks

Yi Cui · Jian Pei · Guanting Tang · Wo-Shun Luk · Daxin Jiang · Ming Hua

Received: 31 October 2011 / Revised: 2 May 2012 /
Accepted: 8 May 2012 / Published online: 2 June 2012

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. 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 paper, such methods may not be effective in practice. To the best of our knowledge, this problem has not been carefully and thoroughly addressed in research. In this paper, we systematically investigate the problem and develop a practical data mining approach. We find that

We are grateful to the anonymous reviewers for their constructive suggestions, which help to improve the quality of this paper. This research is supported in part by an NSERC Discovery Grant, a BCFRST NRAS Endowment Research Team Program project, two SAP Business Objects ARC Fellowships, and two NSERC CRD Research Grants, and a GRAND NCE project. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

Y. Cui · J. Pei (✉) · G. Tang · W.-S. Luk
Simon Fraser University, Burnaby, BC, Canada
e-mail: jpei@cs.sfu.ca

Y. Cui
e-mail: cuiyic@cs.sfu.ca

G. Tang
e-mail: gta9@cs.sfu.ca

W.-S. Luk
e-mail: woshun@cs.sfu.ca

D. Jiang
Microsoft Research Asia, Beijing, China
e-mail: djiang@microsoft.com

M. Hua
Facebook Inc., 1601 Willow Road, Menlo Park, CA 94025, USA
e-mail: arceehua@fb.com

using only the profiles or the graph structures is far from effective. Our method utilizes the similarity between email accounts and social network user profiles, and at the same time 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.

Keywords email mining · social network mining · recommendation systems · string matching · graph matching

1 Introduction

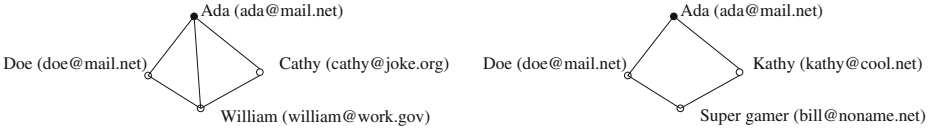
Many people use emails everyday. Emails and the social network formed by email correspondents play an important role in many people's social life. Therefore, many online social networks [19], such as Facebook, LinkedIn, and Twitter, associate users with their emails in one way or another. For example, 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 online social network, such as Facebook. For the sake of simplicity, hereafter we use the term *social networks* to refer to *online social networks*.

Example 1 (Motivation) Figure 1a 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 1b. The problem addressed in this paper 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. 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?

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,



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

Figure 1 A motivation example, where the accounts Cathy and Kathy, William and Super gamer, respectively, are owned by the same persons.

such as "find friends" in Facebook and "search for someone by name" in LinkedIn. Facebook also provides a contact import function, which searches for Facebook accounts whose associated email matches an input email exactly.

Example 2 (Motivation cont'd) In our motivation example (Figure 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".

It is well recognized that matching only based on profiles is far from satisfactory to solve the problem of finding email corespondents in social networks. Our empirical study on two real data sets shows that using profiles only the matching accuracy is lower than 30% (Section 6.3 and Figure 5). 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.

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 engaged 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 facilitate the integration of 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. 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.

Surprisingly, the problem of finding email correspondents in social networks has not been carefully and thoroughly addressed in research. In this paper, we tackle the problem from a practical data mining angle, and make several contributions. First, through an empirical study on two real data sets, we find that using only profiles or graph structures is far from effective to find email correspondents in social networks. Second, 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. Last, 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.

The rest of the paper is organized as follows. In Section 2, we formulate the problem and present the framework of our solution. Section 3, we briefly review the related work. We discuss the profile similarity in Section 4, and develop the neighborhood based similarity and the computation methods in Section 5. We report an empirical evaluation in Section 6, and conclude the paper in Section 7.

2 Problem formulation and framework

In this section, we first formulate the problem of finding email correspondents in social networks. Then, we decompose the problem and present the framework of our solution.

2.1 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 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 paper.

For an account $u \in V_M$, an account $v \in V_M$ is called an **email correspondent** or a **contact**, of u if $(u, v) \in E_M$. 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 (universal social) network** $G = (V, E)$, which captures all relationships among all people. Every person has one and only one vertex in G.

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 $(f(u), f(v)) \in E$. In general, the ground truth network G and the mappings f and g may not be obtainable.

The problem of **finding email correspondents in social networks** is to find a mapping $h : V_M \to V_N$ 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 with respect to a given email account u, assuming $h(u)$, i.e., the social network account corresponding to u, is given. 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, such as an email client. 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 clients 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 paper, 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 practical need in existing social networking sites where more often than not a user is offered a list of recommendations.

2.2 The framework of our solution

As discussed in Section 2.1, we have two types of information in finding email correspondents.

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 Section 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 Section 5.

As to be reported in our empirical study (Section 6.3), using only profile information or graph information can only lead to poor accuracies, less than 30% and 10%, respectively.

Our method integrates profile similarity and graph matching similarity iteratively. The framework of our method has the following two iterative steps.

The profile-based similarity search step 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*.

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

Our method can achieve an accuracy of about 50%, which is substantially higher than those by the profile-only or graph-only methods.

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.

3.1 Email mining

Emails are 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 [20] and Carvalho and Cohen [6] applied clustering, natural language processing and text mining techniques to group email recipients. Balamurugan et al. [2] and Sahami et al. [23] used classification methods to detect spam emails.

Many applications have been developed based on email mining, particularly the email social networks. For example, Roth et al. [22] suggested friends based on an implicit social graph built on email senders, receivers and interactions. Their study led to two interesting Gmail Labs features on contact suggestions. As another example, in the context of a "personal email social network" on people's email accounts, Yoo et al. [29] 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 tackles 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 has been widely used in information retrieval. Many methods have been proposed to capture the similarities between strings. For example, the Hamming distance [12] counts the total number of different characters between two equal length strings. The Jaccard similarity coefficient [21] 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 [17] 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 and Knoblock [18] employed a score that combines the Jaro–Winkler similarity [28], the Jaccard similarity [21] 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 [14], and some others. Elmagarmid et al. [11] 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 [25] proposed subgraph matching algorithms based on tree search. Almohamad and Duffuaa [1] suggested a linear programming approach to the weighted graph matching problem. Van Wyk et al. [26] presented a graph matching algorithm from the functional interpolation theory point of view.

Based on the intuition that two vertices in two graphs are similar if the vertices in their neighborhoods are similar, Blondel et al. [3] introduced a similarity measure for vertices on directed graphs. The directed graphs are represented in their adjacency matrices. A stable similarity matrix X can be obtained by iteratively updating the equation

$$X \leftarrow BXA^T + B^T XA,$$

where A and B are the adjacency matrices of the graphs, respectively, and all entries of the similarity matri X are initialized to 1. This measure was extended to address interests from different aspects.

From the point of view of vertices, in addition to the similarity between the connected neighborhoods, Heymans and Singh [13] considered positive flows from the non-directly connected neighborhoods and penalized flows from the "mismatched" neighborhoods as well. Let I_B be a matrix of all ones and with the same dimensionality as B. The similarity matrix X [30] is computed by

$$\begin{aligned} X \leftarrow\ & BXA^T + B^T XA \\ & + (I_B - B)X(I_A - A)^T + (I_B - B)^T X(I_A - A) \\ & - BX(I_A - A)^T - B^T X(I_A - A) \\ & - (I_B - B)XA^T - (I_B - B)^T XA \end{aligned}$$

In the above formula, the items $BXA^T + B^T XA$ add the scores from the connected neighbors; the items $(I_B - B)X(I_A - A)^T + (I_B - B)^T X(I_A - A)$ add the scores from the non-directly connected neighbors; and items $BX(I_A - A)^T - B^T)X(I_A - A)$ and $(I_B - B)XA^T - (I_B - B)^T XA$ subtract the scores from the mismatched neighbors.

From the point of view of edges, Zager and Verghese [31] brought forward a "vertex-edge" score by using a linear iterative updating framework between vertex similarity and edge similarity.

$$\begin{aligned} Y &\leftarrow B_S^T XA_S + B_T^T XA_T \\ X &\leftarrow B_S YA_S^T + B_T YA_T^T \end{aligned}$$

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.

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

where D_{B_S} and D_{B_T} are the diagonal matrices with the i-th diagonal entry equal to the out-degree and in-degree, respectively, of vertex i.

Many applications use graph matching. For example, Blondel et al. [3] used a similarity score to extract synonyms automatically. Saux and Bunke [24] proposed a graph matching based classifier for image processing.

The idea of graph similarity in this paper is inspired by [3, 13, 15]. As verified by our experimental results on the real data sets (Section 6.4), applying the state-of-the-art graph matching methods directly to our problem does not achieve good performance. Thus, we have to develop our own graph matching method.

4 Profile matching

We discuss the profile matching problem in this section. 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.

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 the similarity of a string 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.

The Jaro–Winkler similarity [28] 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). For example, let s_1 ="abcd" and s_2 ="baec". Then, s'_1 ="abd" and s'_2 ="bac".

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** [14] 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

$$JWScore(s_1, s_2) = JaroScore(s_1, s_2) + (l \cdot p \cdot (1 - JaroScore(s_1, s_2))),$$

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

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.

The overlap similarity [17] is a commonly used string similarity measure. It returns a high score when one string is the substring of the other one.

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 [16] in s.

Example 4 (Overlap similarity) Consider strings "marh" and "marhta". It can be verified that $OvlpScore(\texttt{marh}, \texttt{marhta}) = 1.00$.

The overlap similarity ranges from 0 to 1. A value 0 means two strings are not similar at all; while a value 1 means 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.

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

$$ProfSim(s_1, s_2) = \alpha \cdot JWScore(s_1, s_2) + (1 - \alpha) \cdot OvlpScore(s_1, s_2), \quad (1)$$

where $0 \leq \alpha \leq 1$. We learn the parameter value for α empirically using a set of training data, as described in Section 6.3.

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 profiles and the social network profiles. If a correspondent uses different email addresses and/or names in the profiles, profile matching may not work well.

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, 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 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 1b), we find that Doe and Kathy have a common neighbor "Super gamer". Interestingly, Doe and Cathy in the email network (Figure 1a) 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

$$S^{(0)} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{2}$$

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

$$S^{(i+1)} = f_{\text{iter}}(A, B, S^{(i)}) \tag{3}$$

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 $S^{(i+1)} = AS^{(i)}B$. Their method

is simple, however, assigns very high similarity scores to vertices of high degree [7]. In addition, their method can only be applied to directed graphs. In the case of undirected graphs, the similarity matrix S converges to 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 [27] 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

$$FJ(T_1, T_2) = \frac{FI_{T_1,T_2}}{|T_1| + |T_2| - FI_{T_1,T_2}} \tag{4}$$

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. For example, in the bipartite G in Figure 2, there are multiple possible matchings. Two are shown in the same figure. In bipartite matching m_1, $m_1(a, x) = 1$ and $m_1(b, y) = 1$. $m_1(a, y) = m_1(b, x) = m_1(c, x) = m_1(c, y) = 0$.

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 (4) as

$$FJ(T_1, T_2) = \frac{\sum_{x \in T_1, y \in T_2} s_{x,y} M(x, y)}{|T_1| + |T_2| - \sum_{x \in T_1, y \in T_2} s_{x,y} M(x, y)} \tag{5}$$

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

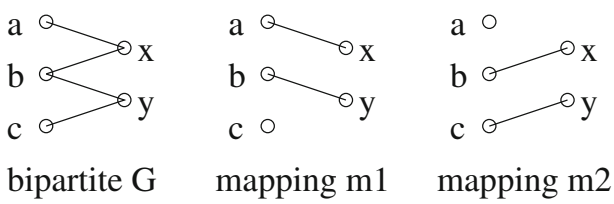


Figure 2 Bipartite matching.

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 (5), we can define the similarity of vertices $s_{v,w}$ as

$$\begin{aligned} s_{v,w} &= FJ(N(v), N(w)) \\ &= \frac{\sum_{x\in N(v), y\in N(w)} s_{x,y} M(x, y)}{|N(v)| + |N(w)| - \sum_{x\in N(v), y\in N(w)} s_{x,y} M(x, y)} \end{aligned} \tag{6}$$

In each iteration, we apply (6) to update each entry of the similarity matrix. We rewrite (6) into an iterative form.

$$\begin{aligned} s^{i+1}_{v,w} &= FJ^i(N(v), N(w)) \\ &= \frac{\sum_{x\in N(v), y\in N(w)} s^i_{x,y} M^i_{v,w}(x, y)}{|N(v)| + |N(w)| - \sum_{x\in N(v), y\in N(w)} s^i_{x,y} M^i_{v,w}(x, y)}, \end{aligned} \tag{7}$$

where $M^i_{v,w}(\cdot)$ denotes the maximum weighted bipartite matching between the neighbor sets of vertices v and w at the ith 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 (2), $s^0_{v,w} = 1$. In the 1st 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

$$\sum_{x\in N(v), y\in N(w)} s^k_{x,y} M^k_{v,w}(x, y) \leq \sum_{x\in N(v), y\in N(w)} s^{k-1}_{x,y} M^k_{v,w}(x, y) \tag{8}$$

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

$$\sum_{x\in N(v),y\in N(w)} s_{x,y}^{k-1} M_{v,w}^{k}(x,y) \leq \sum_{x\in N(v),y\in N(w)} s_{x,y}^{k-1} M_{v,w}^{k-1}(x,y) \tag{9}$$

Combining (8) and (9), we have

$$\sum_{x\in N(v),y\in N(w)} s_{x,y}^{k} M_{v,w}^{k}(x,y) \leq \sum_{x\in N(v),y\in N(w)} s_{x,y}^{k-1} M_{v,w}^{k-1}(x,y). \tag{10}$$

Combining (7) and (10), we have $s_{v,w}^{k+1} \leq s_{v,w}^{k}$.

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

For the sake of efficiency, we can 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. We will empirically evaluate the effect of ϵ in Section 6.6.

5.4 Integrating profile similarity and graph matching similarity

The profile similarity and the graph matching similarity capture different characteristics of email and social network accounts. On the one hand, using profile similarity only may lead to many false negatives. That is, if one provides different profile information in the email and social network accounts, profile similarity cannot match the two accounts. On the other hand, using graph matching similarity only may lead to many false positives, since many small subgraphs in the email and social networks may be similar to each other.

It is natural to use the affine combination of the two similarity scores to improve the accuracy. Formally, for email account v and a social network account w, the overall similarity between them is calculated by

$$Sim^{i}(v,w) = \gamma\, Prof Sim(v,w) + (1-\gamma) s_{v,w}^{i}, \tag{11}$$

where $s_{v,w}^{i}$ 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 similarity scores equals 1. We learn the parameter value for γ empirically using a set of training data, as discussed in Section 6.5.

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, graph matching, and our integrative method.

6.1 Real data set preparation

To test our methods, we need both email data and social network data about individuals. However, 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 not among the authors of this paper agreed to let us test our methods on their email and Facebook data. Under the agreement, the two volunteers must 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 paper.

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 every 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 email profile information.

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 1. While the Facebook 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.

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

Table 1 The statistics of the two real data sets.

	Email network		Social network			
	$\|V_M\|$	$\|E_M\|$	$\|V_N\|$	$\|E_N\|$	$\|V_M \cap V_N\|$	$\|E_M \cap E_N\|$
D_A	2,439	180,524	7,575	440,325	455	2,392
D_B	452	4,676	11,176	566,491	206	736

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** (**accuracy** for short) is thus defined as the percentage of successful matchings. We report the matching accuracy with respect to various answer set size (k).

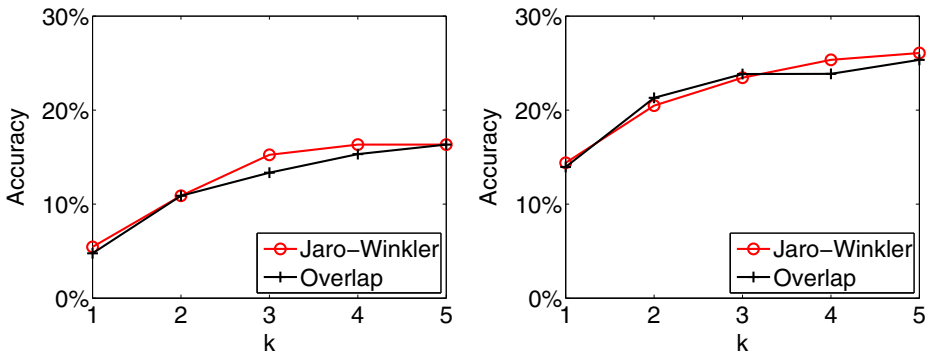
One challenge in evaluation is that even the volunteers do not know the complete ground truth. In other words, the volunteers do not know exactly whether their email contacts have Facebook accounts or not. The volunteers only can determine whether whether a Facebook account belongs to an email contact 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 subsection, 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 Section 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 aggregated to form a profile similarity measure.

Figure 3 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



(a) Dataset D_A (b) Dataset D_B

Figure 3 The matching accuracy of the two string similarity measures.

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 4 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.

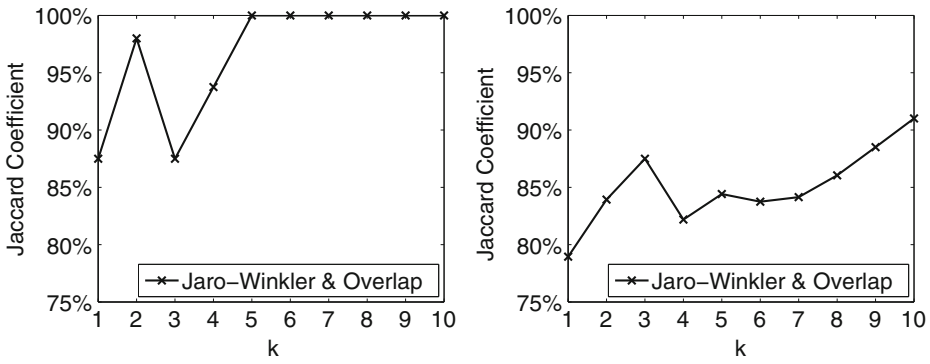
To learn the parameter α in the profile similarity (1), we compute the matching accuracy with respect to α, varying from 0 to 1 of step 0.1. Figure 5 shows the results. For answer set size in the range from 1 to 5, the matching accuracy is the highest when $\alpha = 0.8$. Therefore, we set $\alpha =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

6.4 Effectiveness of graph matching

In this subsection, 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 and Verghese [31] (denoted by ZV) and Heymans and Singh [13] (denoted by HS).

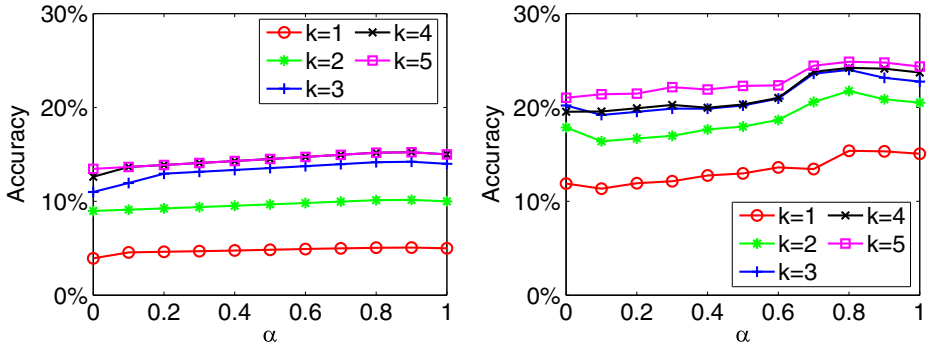
Figure 6 shows the results. The results clearly show that FJ outperforms the existing graph matching methods significantly with respect to a wide range of answer set size. In other words, our graph matching method is dramatically more effective than the other general graph matching methods on the problem of finding email correspondents in social networks.

However, the matching accuracy is still 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.



(a) Dataset D_A (b) Dataset D_B

Figure 4 The Jaccard coefficient between the two string similarity measures in profile matching.



(a) Dataset D_A (b) Dataset D_B

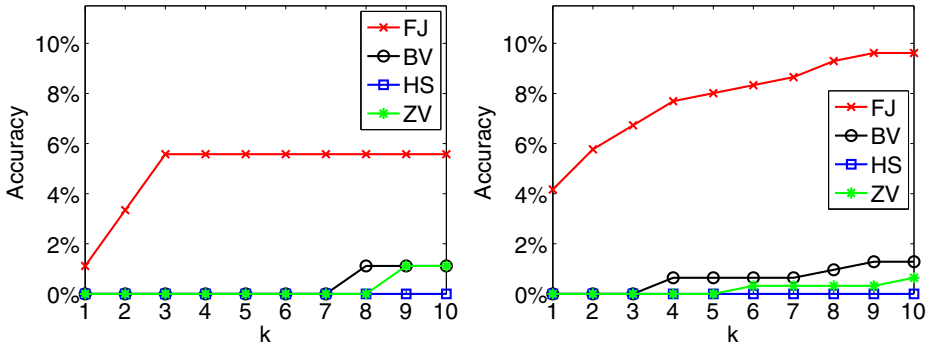
Figure 5 Profile similarity parameter tuning.

6.5 Effectiveness of integrating profile matching and graph matching

In Section 5.4, we developed an iterative method integrating profile matching and graph matching. In this subsection, we report the evaluation results of this approach.

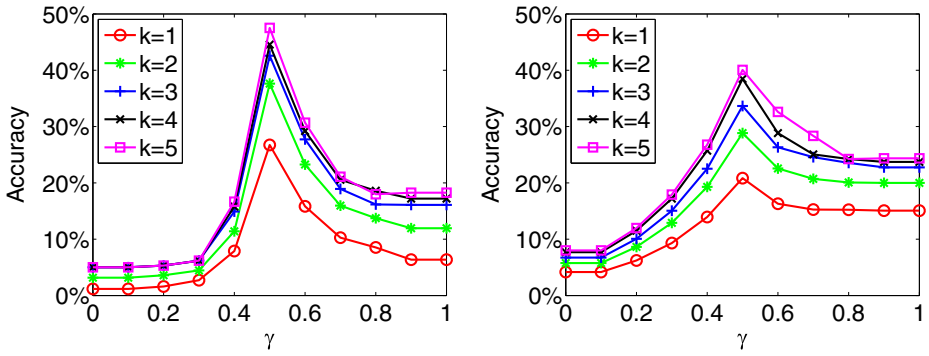
First, we learn the value of parameter γ in (11). Similar to learning the value of parameter α in profile matching, we compute the matching accuracy with respect to γ, varying from 0 to 1 of step 0.1. Figure 7 shows the results. The results show that, empirically in our cases, the best performance is achieved by setting $\gamma = 0.5$.

Figure 8 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



(a) Dataset D_A (b) Dataset D_B

Figure 6 The matching accuracy of the graph matching methods.

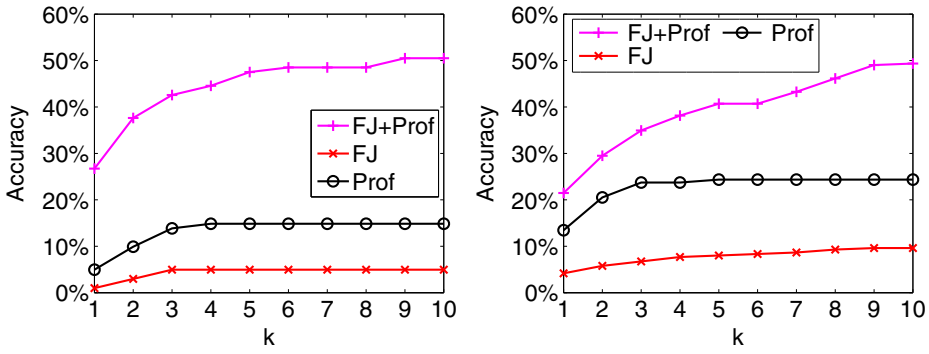


(a) Dataset D_A (b) Dataset D_B

Figure 7 The matching accuracy of the integrated method with respect to parameter γ.

accuracies of graph matching only and profile matching only. The results verify that the integration of the two matching methods iteratively is effective.

One may think that the accuracy of about 50% even when $k = 10$ is not high. Please note that the problem of finding email correspondents is very challenging. In the traditional classification problems where the number of possible classes is very small, some simple baseline methods such as random selection may already give a not bad accuracy. For example, in a balanced two-class classification problem, random selection can achieve an expected accuracy of 50%. However, if the problem of finding email correspondents is treated as a classification problem, the number of possible classes is the number of distinct users in the target social network, and thus is often very large. Thus, the accuracy of about 50% is statistically significant comparing to the random selection method. Our method here considers only email account similarity and social network neighborhood similarity. If more information is available, such as the dynamics of users, the accuracy may be improved further.



(a) Dataset D_A (b) Dataset D_B

Figure 8 Comparison of different matching algorithms.

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 9 shows the accuracy on the two data sets with respect to ϵ. Please note that 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 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.

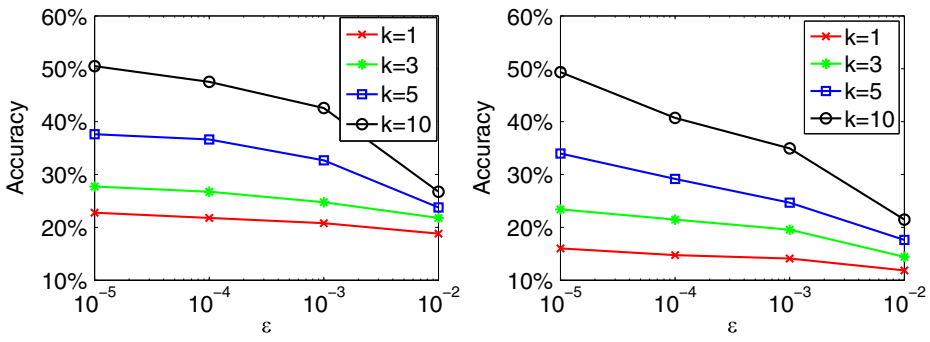
Since the crawling problem is orthogonal to the algorithms developed in this paper, we decided to focus on the runtime cost of the similarity computation. Thus, we report here the runtime that does not count the crawling cost.

Figure 10 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 logarithmic scale. The results clearly show that parameter ϵ can be used to control the tradeoff between accuracy and efficiency.

In our experiments, we compute the huge matrix on the fly. It is known that matrix multiplication is very time consuming. Our method can be sped up substantially by distributed computing adopting similar techniques in many other matrix computation problems. Moreover, to improve the online performance, one can precompute and materialize the matrices. We leave this as a future work.

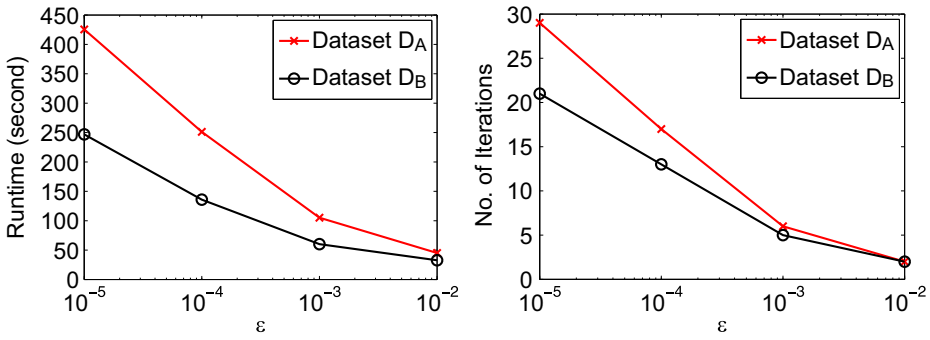
6.7 Distribution of similarity scores and selection using a similarity threshold

To investigate the distribution of similarity scores in the converged similarity matrix computed by our method, we sort all the entries in the matrix in value descending



(a) Dataset D_A (b) Dataset D_B

Figure 9 Accuracy with respect to ϵ.



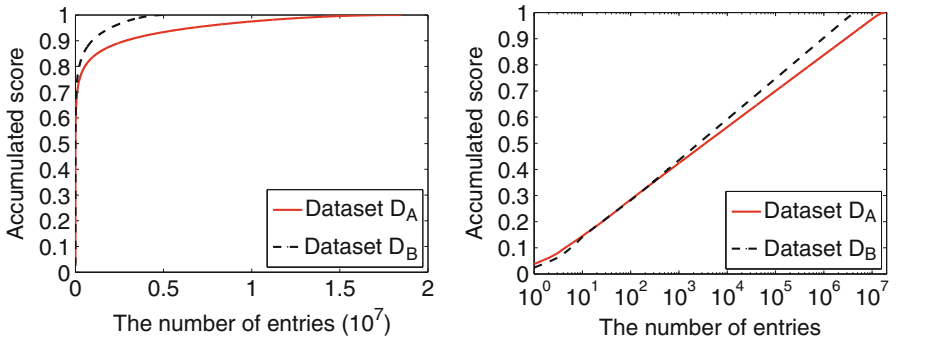
(a) Runtime (b) Number of iterations

Figure 10 Runtime and number of iterations.

order. The size of the similarity matrices for data sets D_A and D_B are $2{,}439 \times 7{,}575 = 18{,}475{,}425$ and $452 \times 11{,}176 = 5{,}051{,}552$, respectively. Since the scores are normalized, in each matrix, the sum of similarity scores of all entries is always equal to 1.

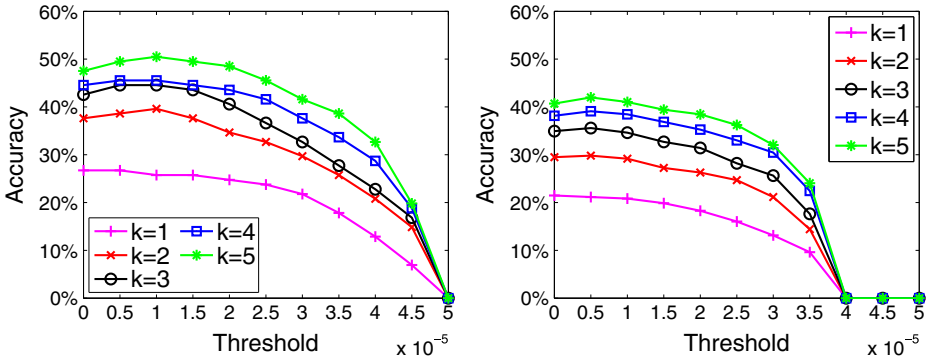
Figure 11 shows the sum of the similarity scores with respect to the top l entries when l varies from 1 to the total number of entries in the matrix. Figure 11a is in the regular scale, and Figure 11b uses the logarithmic scale on the number of entries. As shown, only a small number of entries have high similarity scores. The distribution essentially follows the power law.

Naturally, one may think whether we can use a threshold on the similarity score to make recommendations. That is, we only make recommendations for entries in the similarity matrix whose values pass a user-specified similarity threshold. Figure 12 shows the results on the two real data sets. The results show that the accuracy reaches



(a) Results in regular scale (b) Results in logarithmic scale for number of entries

Figure 11 The distribution of similarity scores.



(a) Data set D_A (b) Data set D_B

Figure 12 Accuracy of recommendation using similarity thresholds.

the peak on those two data sets when the threshold is set to between 0.5×10^{-5} and 1×10^{-5}. When the threshold is low, many entries of low similarity lead to inaccurate recommendation. When the threshold is set too high, many matching vertices may be filtered out.

7 Conclusions and discussion

In this paper, 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.

In this paper, 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 general, this paper is one of the steps of our bigger project on mapping social networks. It is interesting to explore how to carrying one's friends in a social network to many others.

References

1. Almohamad, H.A., Duffuaa, S.O.: A linear programming approach for the weighted graph matching problem. IEEE Trans. Pattern Anal. Mach. Intell. **15**(5), 522–525 (1993)
2. Balamurugan, S.A.A., Athiappan, G., Pandian, M.M., Rajaram, R.: Classification methods in the detection of new suspicious emails. J. Inf. Knowl. Manag. (JIKM) **7**(3), 209–217 (2008)
3. Blondel, V.D., Gajardo, A., Heymans, M., Senellart, P., Dooren, P.V.: A measure of similarity between graph vertices: applications to synonym extraction and web searching. SIAM Rev. **46**(4), 647–666 (2004)
4. Bunke, H.: Graph matching: theoretical foundations, algorithms, and applications. In: Proc. Vision Interface, pp. 82–88 (2000)
5. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning graph matching. IEEE Trans. Pattern Anal. Mach. Intell. **31**(6), 1048–1058 (2009)
6. Carvalho, V.R., Cohen, W.W.: Preventing information leaks in email. In: SIAM Data Mining Conference (SDM) (2007)
7. Cason, T.P., Absil, P.-A., Dooren, P.V.: Review of similarity matrices and application to subgraph matching. In: Book of Abstracts of the 29th Benelux Meeting on Systems and Control, p. 109 (2010)
8. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: IIWeb, pp. 73–78 (2003)
9. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. Pattern Anal. Mach. Intell. **26**(10), 1367–1372 (2004)
10. Dice, L.R.: Measure of the amount of ecologic association between species. Ecology **26**(3), 297–302 (1945)
11. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: a survey. IEEE Trans. Knowl. Data Eng. **19**, 1–16 (2007)
12. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)
13. Heymans, M., Singh, A.: Deriving phylogenetic trees from the similairty analysis of metabolic pathways. Bioinformatics **19**(suppl 1), 138–146 (2003)
14. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. J. Am. Stat. Assoc. **84**(406), 414–420 (1989)
15. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM) **46**(5), 604–632 (1999)
16. Kondrak, G., Marcu, D., Knight, K.: Cognates can improve statistical translation models. In: North American Chapter of the Association for Computational Linguistics - Human Language Technologies (HLT-NAACL) (2003)
17. Lawlor, L.R.: Overlap, similarity, and competition coefficients. Ecology **61**(2), 245–251 (1980)
18. Michelson, M., Knoblock, C.A.: Semantic annotation of unstructured and ungrammatical text. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 1091–1098 (2005)
19. Musial, K., Kazienko, P.: Social networks on the internet. World Wide Web: Internet and Web Information Systems (2012). doi:10.1007/s11280-011-0155-z
20. Pal, C.: Cc prediction with graphical models. In: 3rd Conference on Email and Anti-Spam (CEAS) (2006)
21. Paul, J.: *Étude* comparative de la distribution florale dans une portion des alpes et des jura. Bull. Soc. Vaud. Sci. Nat. **37**, 547–579 (1901)
22. Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Matias, Y., Merom, R.: Suggesting friends using the implicit social graph. In: 16th ACM SIGKDD Conference on Knowlegde Discovery and Data Mining (KDD), pp. 233–242 (2010)
23. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approcah to filtering junk e-mail. In: Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Learning for Text Categorization (1998)
24. Saux, B.L., Bunke, H.: Feature selection for graph-based image classifiers. In: Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA) (2), pp. 147–154 (2005)
25. Ullmann, J.R.: An algorithm for subgraph isomorphism. Journal of the ACM (JACM) **23**(1), 31–42 (1976)
26. van Wyk, M.A., Durrani, T.S., van Wyk, B.J.: A rkhs interpolator-based graph matching algorithm. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 988–995 (2002)

27. Wang, J., Li, G., Feng, J.: 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)
28. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proceedings of the Section on Survey Research, pp. 354–359 (1990)
29. Yoo, S., Yang, Y., Lin, F., Moon, I.-C.: Mining social networks for personalized email prioritization. In: 15th ACM SIGKDD Conference on Knowlegde Discovery and Data Mining (KDD), pp. 967–976 (2009)
30. Zager, L.: Graph Similarity and Matching. Master's thesis, Massachusetts Institute of Technology, USA (2005)
31. Zager, L.A., Verghese, G.C.: Graph similarity scoring and matching. Appl. Math. Lett. **21**(1), 86–94 (2008)