User Identification across Social Networks using the Web Profile and Friend Network

Jan Vosecky, Dan Hong, Vincent Y. Shen
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong
jvosecky@gmail.com, csdhong@ust.hk, shen@ust.hk

ABSTRACT: *Today, it is common that people are users of more than one social network and their friends may also be registered on multiple websites. A facility to aggregate our online friends into a single integrated environment would enable us to keep up-to-date with our virtual contacts more easily, as well as to provide improved facility to search for users across different websites. In this paper, we propose a method to identify users based on web profile matching and further extend its effectiveness by incorporating the user's friend network. We collect and study real-life data from two popular social networks and evaluate the importance of profile information. Machine learning algorithms are used in our experiments and we present results of pure profile-based user identification and demonstrate the benefits of incorporating the friend network in the classification process. We show that our combined method successfully identifies up to 93% of duplicated users across social networking websites.*

Keywords: Social networks, User profile, Machine learning, Privacy

Received: 12 August 2009, Revised 21 September 2009, Accepted 18 October 2009

1. Introduction

After the search engine, social network is another hot Web application which involves the user as the main actor. The success of social networking websites, such as Facebook, MySpace, and others, shows the revolution caused by Web 2.0. Users have their "second life" on the Web, which is a virtual environment to meet friends, discuss opinions, play online games and share information. It also is more and more common that users are members of more than one social network at a time and that their friends similarly use multiple social networks.

Currently, different social networking websites use different ways to store and display information about a user - user's Web profile. Figure 1 shows how different websites display similar profile information. Also, due to business and privacy concerns, services from different providers communicate only by providing basic import functions which should be manually driven by the users. As a result, the users' preferences for different social networking websites make the exchanging of friends' information difficult.

For a user of multiple social networks, there is increased overhead in keeping up-to-date with friends' activities. There is also increased complexity in choosing the "right" communication platform; that being simply the one used by a particular friend of the user at a particular time. Taken from a different perspective, people searching becomes another challenge. As people are registered on one or more websites, performing a simple search across a number of social networking websites only produces separate (but possibly overlapping) results for each social network domain.

The method presented in this paper may help to tackle these challenges. An application, for example, would be in creating a Web-based people search facility. This facility could be used to search for individuals in a number of social networks and produce a consolidated result with duplicated user profiles identified and their profiles "summarized". Another application might be a "meta" social network website which provides a single environment for users to access their virtual lives. Users could link their accounts from different social networks and the meta social

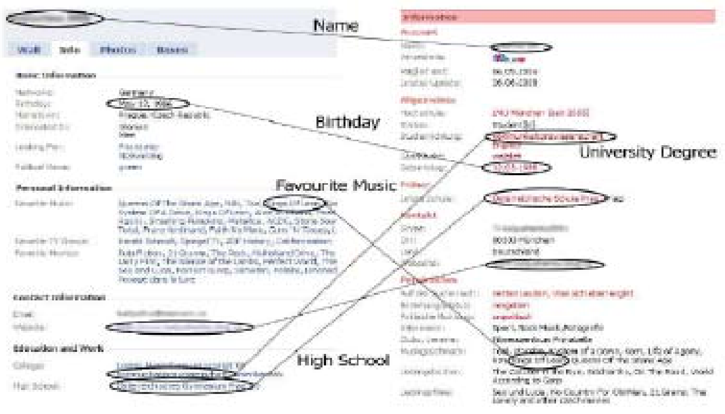


Figure 1. Two profiles of same user on two websites with matching details

network website would consolidate all their details and friends' profiles. In this way, the user would have a simple and effective way of keeping up-to-date with their friends' activities and communicating with them across all social networks from a single environment.

In this paper, we present a method of matching web profiles from different domains to identify whether they belong to the same real person. Our proposed method uses the representation of a profile as a vector, consisting of individual profile fields. We may then calculate a similarity score between two profile vectors. Because of the different kinds of information stored on a user profile (compare 'user name' and 'favorite movies'), we describe methods to match different profile fields. We focus on the differences between profile fields and their importance in the matching process. We use real-life data collected from two popular social networks to examine which profile fields are most important for user identification.

To extend the profile matching method, we then present a structural approach to identifying a user based on their friend-network. We calculate a mutual friends-overlap between two users on different domains to enhance the identification process. We combine both methods to improve user identifications in cases when there is knowledge about the friend-network, but little or no profile information is available.

For our experiments, we employ machine learning algorithms to classify our crawled data and we present results with different machine learning methods. Our experiments show that using the Web profile and friend- network, we can achieve more than 90% of successful identifications of duplicated users across domains.

2. Related Work

The approach of calculating a similarity score between a pair of entities has been studied and applied in a variety of areas in the past, including string similarity (or distance) [2, 4], document similarity used in document clustering [1] and information filtering [12]. These approaches have been extended into further applications, such as genetics, natural language processing, image processing and others [13, 20].

Vector-based comparison algorithms have been used for document and query matching in traditional search engines [14, 19] and more recently in ontology-based search engines [3]. The notion of user profile vector comparison has been introduced in relation to collaborative information retrieval (CIR) systems [16, 17, 18]. These methods remain focused on the query-matching/document-retrieval problems. In contrast, our work focuses on social network user profiles. The profile matching problems together with the graph-based algorithm introduced in [16] are effectively handled in our approach by our field matching techniques and the vector-based comparison algorithm incorporating weights.

Work has been done in the field of personal information reconciliation [7, 6], which includes identifying duplicated person references among personal data (documents). The stepwise comparison method proposed in [7] may be applied to social networks, however our approach allows for more flexibility and complexity, which is important in profile comparison on social networks. A general reference reconciliation method is proposed in [5]. The difference of our work is the special focus on profile matching in social networks and the empirical study of the importance of profile fields.

Recent effort has been made on social network integration, also referred to as social network "mashup". An ontology-driven social network integration experiment in [21] focuses on relationships between colleagues; however, it only considers the persons' first and last names for matching. In our work, we consider a wide range of profile fields and investigate their importance for matching. This can increase the success of user identification, even in cases of missing or ill-formed information, as well as reducing the possibility of misidentifications due to two users sharing the same name.

Work has been done on structural analysis of the social network graph. Hsu et al. [11] present methods to analyze the social graph for demographical and community knowledge. Hay et al. [10] present a method of vertex re-identification in an anonymized graph, which we implemented in our project.

3. Comparing Profiles

3.1. Vector-Based Profile Matching

In our matching method, each user profile is represented as a vector. Similar to the approach used in Web search engines [14, 19], which represents websites as vectors consisting of document words, a profile vector represents profile data fields (e.g., phone number, date of birth, etc.). An n-dimensional vector would represent n different profile fields for a particular person.

Definition 1 *A profile vector P is defined as $P = <f1, f2, \ldots, fn>$, where fi is the ith profile field.*

For example, a profile vector representing 3 fields ($n = 3$) would be: *P1* = < nickname, email, date of birth >. Comparison of any two vectors *P1* and *P2* consists of two stages. In the first stage, the algorithm utilizes string matching functions to calculate a similarity score between corresponding vector fields. A similarity vector is obtained as a result of this operation.

Definition 2 *A similarity vector V is defined as $V(P1, P2) = < v1, v2, \ldots, vn >$, such that $vi = compi(fi{,}P1, fi{,}P2)$, for each vi, $0 \le vi \le 1$, $|V| = |P1| = |P2|$*

The function *compi* is a chosen field comparison function that takes data in field *fi,P1* from *P1* and *fi,P2* from *P2* $(0 < i \le n)$ and returns a value between 0 and 1. In general, $vi = 1$ if *fi,P1* and *fi,P2* are identical; $vi = 0$ if there is no similarity between *fi,P1* and *fi,P2* . The function *compi* may be different from field to field since each field may have a different format. For example, a comparison of gender returns an integer value 0 or 1 depending on a match; however, comparing addresses may produce a rational number.

Problem 1 *How do we compare complex profile fields during stage 1 of the classification process?*

If a profile field contains a complex data type instead of a string (e.g., address), we may apply the vector comparison approach recursively to calculate the similarity between two complex data types. Then, we may use the resulting similarity for the similarity vector V.

3.2. Matching Profile Fields

For the purpose of the vector comparison algorithm, we distinguish 3 categories of profile field matching functions based on their matching method. In this section, we introduce the 3 categories and give examples of functions we have used in our experiments. Since the algorithm may use any chosen field matching functions, the examples given only illustrate the types of functions that may be employed.

3.2.1 Exact Matching

The first category of matching functions uses simple comparison operations to check equality of data fields. Matching functions of this type produce a boolean result. For example, we may use an exact field matching function to match "gender".

3.2.2 Partial Matching

Partial matching functions allow matches of parts of data fields. They are useful in cases where the user-entered data contains abbreviations, misspellings or in the case of missing data in a complex data type (e.g., address). The following is a multi-purpose substring match function:

$$Sub\,Match(s_1,s_2)=\begin{cases}\frac{length(s_2)}{length(s_1)}\,if\;s_1\text{contains}\,s_2\\ \frac{length(s_1)}{length(s_2)}\,if\;s_2\text{contains}\,s_1\\ 0\quad otherwise\end{cases} \tag{1}$$

This function might be used from within another partial matching function to enable matching of more complex data types. An example of such a function is address matching. The following function allows for partial matches of addresses. As mentioned in Problem 1, this function takes a vector representation of the complex data field “address” as input to calculate a similarity.

$$VMA(A_1,A_2)=\sum_i^n w_i\times Sub\,Match(f_{i,A_1},f_{i,A2}) \tag{2}$$

where $f_{i,A1}$ and $f_{i,A2}$ are the i_{th} field of the address vector A_1 and A_2 respectively, w_i is i_{th} weight of address field f_i and $\sum_i^n w_i=1$.

3.2.3 Fuzzy Matching

When analyzing certain profile fields, we may require more complex logic when calculating a similarity score. When comparing two users’ names, for instance, we need to deal with cases of initials, shortened forms, special characters, etc. In these cases, simple substring matching might not produce the desired results.

For our work, we have designed the *MatchName (MN)* function for comparing two users’ names. It handles full and partial matches of names consisting of one or more words. The *MN* function operates in two stages: preprocessing and matching. In the pre-processing stage, special symbols such as “*$#.” are removed from the input string. Also, specific “black-listed” words are removed from the input string. These words (including “Facebook” and “bitte”) occur in a number of user names on the StudiVZ website. In the second stage, a matching function is applied, which has the following features:

Each word in the name is matched separately. This supports the case of swapped names (surname first vs. given name first); e.g., *MN*(“John Doe”, “Doe John”) = 1

In cases of partial matches, *MN* uses an approach of “counting” the number of matched words:

If the name consists of 2 words and one of them matches, the score will be 1/2 = 0.5

Partial matches are supported. A partial match can be obtained from the longest common substring [15] (LCS) of at least 3 letters, or from a match of initials, whichever score is higher. Partial match score is added to the total score.

Similarity between two names is calculated using the formula:

$$MN(w_1,w_2)=\frac{\sum_{k=1}^{|w_2|} max\,w_2[l]\in w_2\,(Part(w_1[k],w_2[l]))}{\max(|w_1|,|w_2|)} \tag{3}$$

where w_1 and w_2 are arrays of individual words from *name* 1 and 2, $|w_i|$ is the number of words in w_i, $w_i[k]$ is the *k*-th word from array w_i. The formula uses a support function *Part* to calculate a partial string match:

$$Part(s_1,s_2)=max\begin{cases}1 & \text{if}\;s_1=s_2\\ 0.5 & \text{if}\;s_1\,\text{is an initial of}\;s_2\\ 0.5 & \text{if}\;s_2\,\text{is an initial of}\;s_1\\ \frac{LCS(s_1,s_2)}{max[l(s_1),l(s_2)]} & \text{if}\;LCS(s_1,s_2)\geq 3\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

where *s1, s2* are input strings, *l(s)* is the length of string *s, LCS* is a function returning the longest common substring.
In contrast to existing string matching functions based on string distance, token distance and other metrics [2], *MN* provides more control over the result when matching user names. This is mainly because of its specific design for user names in social networks. Table 1 gives examples of the similarity scores given by *MN* and the edit-distance based algorithms by Jaro-Winkler (*J-W*) and Monge-Elkan (*M-E*) [4].

	String Pair	**MN**	**J-W**	**M-E**
1	"John Doe", "Doe John"	1	0	0.5
2	"John Doe", "J Doe"	0.75	0.1	0.8
3	"John Doe", "J… Doe"	0.75	0.78	0.5
4	"John Doe", "P Doe"	0.5	0	0.8
5	"John Novak", "John Nov@k"	0.8	0.96	0.84
6	"Mary Doe", "**Mary Doe**"	1	0	1.0
7	"John Richter", "John FACEBOOK Richter"	1	0.81	0.67
8	"John Doe", "Peter Smith"	0	0.31	0.15

Table 1. String Match Functions Comparison

Table 1 presents similarity scores given by string matching functions in the following cases:

- Swapped names (instance 1): only *MN* gives the desired score of "1", *J-W* and *M-E* give a very low score.
- Use of initials (instance 2-4): *MN* gives a consistent score "0.75" when initials match, whereas *J-W* and *M-E* may give a very high or very low score. When initials don't match (instance 4), *MN* gives score "0.5" since only 1 out of 2 words match. *J-W* gives "0" score and *M-E* gives a very high score, both of which are not desirable.
- Use of a special character (instance 5): here the result of *MN* is similar to *M-E. J-W* gives an even higher score.
- Excessive symbols (instance 6): due to input pre-processing, *MN* and *J-W* give the desired score "1". *M-E* gives a score of "0".
- Additional words in user's name (instance 7): due to input pre-processing, *MN* gives the desired score "1". *J-W* comes close with a high score, but *M-E* gives a much lower score.
- Names completely different (instance 8): MN gives the desired score "0", whereas both other functions give a positive score (*J-W* gives "0.31").

As can be seen from the comparison above, in the cases of swapped names, partial matches and matches of initials, *MN* gives more coherent and "common-sense" scores than the other string matching functions. Due to input preprocessing, *MN* also achieves better results if a name contains special symbols or words. Due to these reasons, we chose *MN* to use for name-matching in the social network domain.

3.3 User Identification

After obtaining the similarity vector *V* introduced in 3.1, the last step is to determine the classification label ("same user" or "different users") for a pair of profiles *P1* and *P2*.

We may employ different techniques to obtain a classification label. The simplest method is to use a weighting vector *W* of the same dimension as *V* in order to control the influence of each profile field. We may then calculate a product of *V* and *W* to obtain a similarity score:

$$S(P_1, P_2) = \sum_{i}^{n} (v_i \times w_i) \tag{5}$$

The similarity score *S* may then be compared against threshold *t* in order to determine the classification label.

Alternatively, a more systematic solution is to employ machine learning algorithms to build a classification model. We may obtain a classification model from a training dataset and use the model to classify new similarity vectors. In our work, we evaluated several machine learning methods and explored their classification effectiveness. One of the evaluated methods is a multi-layer perceptron (MLP) [8] classifier, which uses the profile vector dimensions as inputs and produces a classification label as output. A multilayer perceptron is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate output data. We trained an MLP model using a dataset of profile similarity vectors, automatically determining the weights of each profile field.

Testing results using a simple weighting vector and more complex machine learning algorithms, including an MLP, are detailed in Section 4.

3.4 Incorporating the Friend Network

As an extension of the profile matching approach, we may also include the relationships between users in the user identification process. Structural similarity will become particularly useful in cases when information on the user profile is missing, incomplete or unavailable. The following section presents an extension to the vector-based profile matching approach by including a structural similarity score.

In our work, we have designed an algorithm based on the recursive formula from [10] with 1-level neighbor matching, corresponding to the immediate friends of a user. We calculate a mutual friends' overlap (MFO) between users on two social networks. For a given pair of users in two domains, the algorithm matches both users' friend-lists to calculate the total number of mutual friends. This has been done by matching the friends' names using the *MN* function introduced in Section 3.2.3 and adding up all similarity scores above 0.75 to obtain a total friends overlap score. The minimum name similarity of 0.75 was chosen so that only names that are highly similar will be considered for the MFO score. Figure 2 illustrates the MFO score calculation. MN score represents similarity given by the *MN* function.

Intuitively, a high mutual friends overlap means that users x and y share a lot of friends. It is likely that x and y know each other in real life, or that they in fact are the same person. In our approach, a high mutual friends overlap means that the prob-

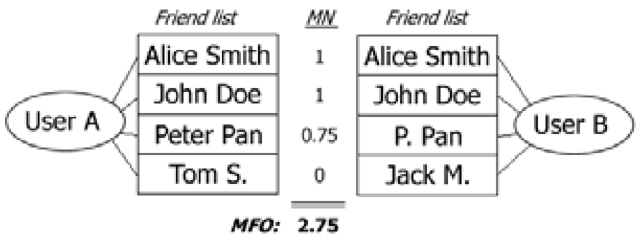


Figure 2. Calculating a mutual friends overlap (MFO)

ability that x and y are identical increases.

In our experiments, we have included the structural similarity score in the similarity vector *V* introduced in 3.1. Results of using this method to extend the profile based similarity method are presented in Section 4.

4. Experimental Evaluation

In this section, we will evaluate the results of our profile matching methods to discover identical users in different social network domains. We will first describe the datasets used for evaluation. Then, we will describe how a set of profile fields for evaluation has been determined. In the next sections, we will describe classification methods used, our experiment methodology and finally, we will present testing results.

4.1 Datasets

In this section, we describe the datasets used in our experiments. For the purpose of this research, we have used two separate datasets: a training dataset and a testing dataset. Both datasets have been obtained by the method of crawling real user profiles from popular social networking websites Facebook and StudiVZ (a German-language social network). Both datasets have been manually checked for overlapping users before the evaluation took place. Methods for manual checking were: checking for similar names, checking of known identical users, as well as matching other profile details (including personal website, date of birth, etc.) to list all potentially identical profile pairs. These potentially identical users were then manually confirmed.

Since the purpose of our research is user identification based on profile matching and friend network matching, we have only considered users with an available profile and/or friend list. Users not sharing any profile information or friend list were not considered as candidates for identical users. Table 2 shows the size of the training and testing datasets crawled from each website and the number of identical users that have been confirmed present on both websites. The following sections discuss how each dataset has been obtained.

	Training DS	**Testing DS**
Facebook	1000	1000
StudiVZ	1000	1000
Identical users	21	43

Table 2. Training and Testing Datasets

4.1.1 Training Dataset

The training dataset has been obtained by crawling 2000 user profiles and corresponding friend-lists from the Facebook and StudiVZ websites. The crawling process has been done in a breadth-first search order, starting from the same user who is registered on both websites.

4.1.2 Testing Dataset

The testing dataset has been obtained by "directed" crawling of specific users and their friends in order to collect a higher number of overlapping users between Facebook and StudiVZ. This approach has been chosen due to the relatively low number of overlapping users obtained in the training dataset. For the training dataset, this has been caused by the use of a breadth-first search approach which has produced a bias in the crawling direction and produced relatively disjoint datasets. For the testing dataset, we have crawled friends of specific users who are present on both Facebook and StudiVZ. After that, we have identified a number of Facebook profiles from the German-speaking territory, searched them on StudiVZ and crawled those profiles.

4.2 Selecting Vector Dimensions

For the purpose of the vector-based profile matching method, we proceeded to select a suitable set of profile fields to consider for our evaluation. Rationale for selecting vector dimensions has been their ability to influence the classification outcome. For this purpose, we used the training dataset and measured statistics about each profile field's presence in the dataset. Figure 3

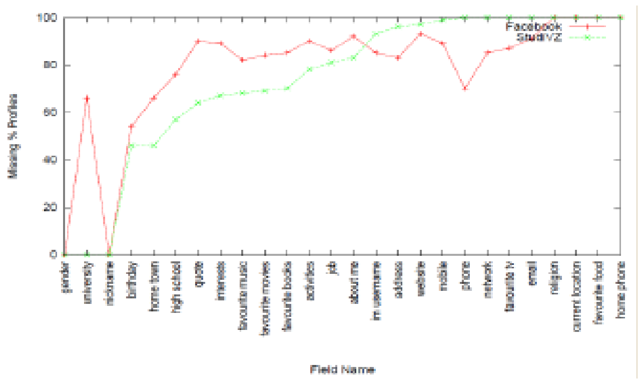


Figure 3. Proportion of profiles with missing fields

shows a list of profile fields and the corresponding percentage of profiles in the dataset which do not include this information. The vector dimension was then chosen to be the first six profile fields in the list (i.e., nickname, gender, birthday, home town, university, high school). In addition to that, we have also included unique identifiers, such as “instant messenger username” and “website”. This information is present in some profiles and may lead to accurate user identification. Therefore, the resulting vector dimension we used is 8.

4.3 Experiment Methodology

4.3.1 Classifiers

This section describes the classifiers used in our experiments. As mentioned in Section 3.3, there are a number of methods available that can be used to classify profile similarity vectors. An example of similarity vector data is given in Table 3.

V = < nickname	**gender**	**birthday**	**home town**	**university**	**high school**	**IM username**	**website**	**friend-list >**	**Class**
Example data:									
V_1 = < 1	1	1	0.24	0	0	0	0	9.16 >	Identical
V_2 = < 0.5	1	0	0	0	0	0	0	0 >	Different

Table 3. Similarity vector definition used for experiments

Initially, we described a simple method of classification based on calculating a similarity score. For our experiments, we have implemented a classifier using the formula in Section 3.3. This classifier is referred to in our experiments as Simple-weights. The value of threshold t chosen for our evaluation is t = 3. To choose values of the weighting vector W, the following approach was used: Firstly, we selected initial weights. Rationale for selecting initial weights was intuitive consideration[1]. Using the selected weights, we performed tests when changing each individual weight while keeping other weights fixed. The rationale was to maximize the success rate. As a result, we determined a suitable weighting vector to use in our tests[2].

To employ a more systematic classification process for our evaluation, we investigated a number of machine learning methods. As such, there is a large number of machine learning algorithms that could deal with our dataset, including, but not limited

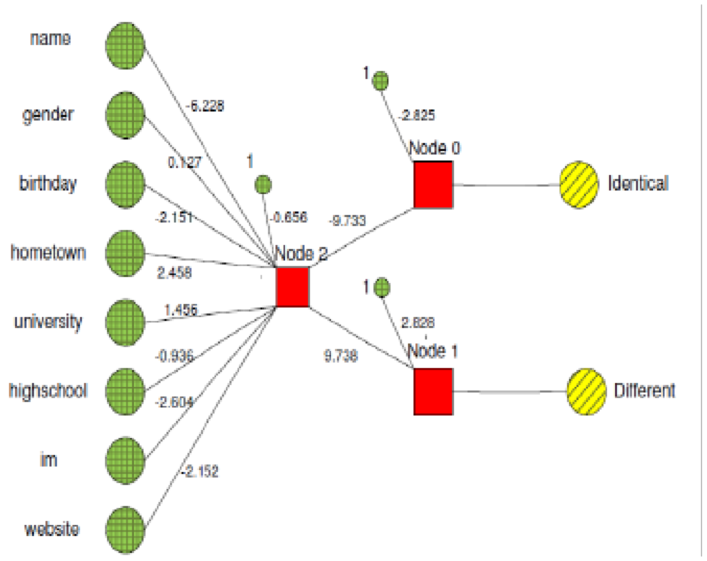


Figure 4. Illustration of MLP classifier used in experiments

[1] The following example illustrates our intuitive consideration: “email address” is a unique identifier, so it should be enough evidence for a successful user identification. Hence the weight of “email” is equal to the threshold value 3. However, “birthday”, for example, is not a unique identifier and we need other information to support the similarity score, such as “name”. Hence the weight of “birthday” is 2 and weight of “name” is 1.4. Together they are enough to reach threshold *t*.

[2] The weights used are as follows: *W*(IM) = 3, *W*(Website) = 3, *W*(Name) = 1.4, *W*(Home Town) = 0.5, *W*(Birthday) = 0.5, *W*(University) = 0.5, *W*(High School) = 0.5, *W*(Gender) = 0.25

Learning rate	0.3
Momentum	0.2
Hidden layers	1
Num of epochs	200
Random seed	0

(a) Parameters - MLP

Logistic: ridge	1.0E-8
Logistic: max. iterations	-1
AdaBoost: iterations	10
AdaBoost: seed	1
AdaBoost: weight threshold	100

(b) Parameters – AdaBoost w/ Logistic

LinReg: attribute selection	M5 method
LinReg: ridge	1.0E-8
LogitBoost: likelihood threshold	-1.798
LogitBoost: iterations	10
LogitBoost: seed	1
LogitBoost: shrinkage	1.0
LogitBoost: weight threshold	100

(c) Parameters – LogitBoost w/ LinReg

Table 4 (a-c). Initial parameters for classifiers used in experiments

to, Naïve Bayes, artificial neural networks, rule-based prediction, ensemble learning, etc. A comprehensive evaluation of available algorithms is, however, outside the scope of this research. Therefore, we chose three representative methods that we found most effective for our classification purposes and compared their performance when dealing with our datasets.

Firstly, we employed a multi-layer perceptron (MLP) [8] classifier as mentioned earlier. Figure 4 illustrates a MLP with 1 hidden layer used in our experiments and how profile fields are automatically assigned weights.

Secondly, we chose an additive logistic regression method (LogitBoost) [15], which performs classification using a regression scheme as the base learner. As the base learner, a simple linear regression model was used.

Thirdly, we chose an Adaptive Boosting (AdaBoost) [15] classifier in conjunction with a multinomial logistic regression model (Logistic). AdaBoost is a meta-classifier for boosting the performance of a nominal class classifier. In other words, a number of classifiers are built simultaneously and their results combined. We used AdaBoost in conjunction with a multinomial logistic regression model (Logistic). This method proved particularly effective in our experiments as can be found in Section 4.4.

The initial parameters that were used with each of the machine learning classifiers are detailed in Table 4.

4.3.2 Experiment Description

In this section, we describe how our experiments were conducted. Our experiments were done on two different levels: profile only and profile with friend-list. Firstly, we performed tests with the user profile only. These tests used similarity vectors as described in Section 3.1 and the vector dimension as described in 4.2. Secondly, we incorporated the structure-based similarity using the user's friend-list, as described in Section 3.4. These tests used similarity vectors with the same dimension as previously and in addition included the structure-based similarity value. Our goal was to investigate the effect of including the friend-list in the classification process.

Experiments consisted of the following four stages: structure-based similarity computation, similarity vector computation, data pre-processing for classification and classification. We will describe each stage in turn. In the first stage, we computed structure-based similarity between the Facebook and StudiVZ datasets for both the training and testing datasets. In the second stage, we computed similarity vectors for both the training and testing datasets. Table 3 gives an illustration of similarity vector data. In the next stage, similarity vector data could be directly passed to the classifiers described in Section 4.3.1. However, preliminary tests showed that classification models built on our training dataset were unsatisfactory. The main problems were (1) bias towards a specific profile field ("name", in particular), resulting in misclassifications, (2) too strict models, resulting in a low recall rate with the testing dataset. As we found, the training dataset has more complete profile and friend-list information than the testing dataset. In the testing dataset, a number of profiles have no friend-list available or no profile information except the user's name. To tackle these problems, we introduced a new pre-processing stage with respect to the training dataset, before actual classification took place. In this stage, the similarity value of the field "name" was reset to "0" in 1/3 of "identical" user's similarity vectors. This helped to tackle problem (1). For tests that use profile with friend-list, the structure-based similarity value was also reset to "0" in 1/3 of "identical" user's similarity vectors. This helped to tackle problem (2).

Lastly, in the classification stage, we performed tests using both the profile-only based classification and the profile with structural-similarity classification. The testing results with different classifiers during this stage are presented in the next section.

4.4 Experiment Results

In this section, we present experiment results with testing data, using the methodology described in the previous sections. The main focus of our experiments was to measure the effectiveness of a particular classifier in correctly classifying identical users across the domain pair. Here, we have a few remarks about the testing dataset, which will help to understand the testing results. The total number of identical users in the testing dataset, detailed in Table 2, comprises of three main types of data available: (1) users with profile information and friend-list available, (2) users with profile information only and (3) users with only a name and friend-list, but no profile available. We believe that these scenarios commonly occur in real-life situations when dealing with social networking data. Hence, our ultimate goal is to retrieve all identical users, regardless of which one of the three types they are. As we can see from the results, different classification methods are effective for different data available to us.

Table 5 presents classification results for the two user identification approaches: profile-based identification and identification incorporating the friend-list in the similarity vector. Numbers in the table correspond to recall rates with respect to identical users in the testing dataset, i.e. users who are registered on both social networking domains.

Method	**Profile only**	**Profile w/ Friend list**
Simple-weights	67.4%	-
AdaBoost w/ Logistic	74.4%	93%
MLP	83.7%	88.4%
LogitBoost w/ LinReg	74.4%	81.4%

Table 5. Testing classification results: identical user recall

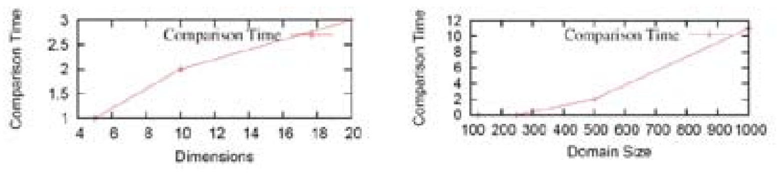
As described in Section 4.3.1, we used four different classifiers in our tests. The Simple-weights classifier was only used in the early stages of this project and we are including it here for illustration purposes.

4.5 Results Discussion

From the results in Table 5, we can observe that incorporating user's friend-lists in the identification process increases successful classifications. As we mentioned in Section 4.4, there are three types of user data available and we can see that the "profile only" method generally only identifies type (1) and (2) user data. In the case of type (3) user data, the "profile only" method would work only if the weight of the user's name was very high. However, as noted in Section 4.3.2, this bias leads to misclassifications when two users share a very similar (or same) name. Classification using the profile and friend-list, on the other side, can deal with all three types of user data.

From the three machine learning methods, the AdaBoost classifier with logistic regression had the best overall result with 93%. The MLP classifier with 1 hidden layer had the best results for profile-only classification and relatively high results for profile with friend-list classification. Both LogitBoost and AdaLine had similar results for profile-only classification, but the LogitBoost model was the strictest with respect to the testing dataset.

In our tests, we used the same input data and hence the same starting conditions for each classifier. We believe that for all classifiers, success rates could be further improved by fine-tuning each individual classifier and/or further data pre-processing to build better classification models.



(a) Running time with respect to domain size

(b) Running time with respect to vector dimension

Figure 5. Performance evaluation

4.6 Performance Evaluation

To evaluate the performance of the comparison algorithm, we have performed a series of tests with the training dataset, using the Simple-weights classifier and 8 vector dimensions. The results of testing with different sizes of the profile domain are presented in Figure 5(a).

From the information in Figure 5(a), we may see that the comparison algorithm operates in the time complexity nearing O(m×n), where m is the number of profiles from Domain A and n is the number of profiles from Domain B. Apart from domain size, performance time also depends on the number of vector dimensions, as it may be seen from Figure 5(b).

5. Conclusions

In this paper, we present a method of Web profile matching in order to determine whether two Web profiles relate to the same real-life person. Further, we extend this approach by utilizing the friend-network of a user and introduce a structure-based similarity measure. We then combine both methods to build a profile-matching tool that uses profile field matching and structure-based matching. We use Web profiles from two different social networking websites and study fields in the profiles which are suitable for the profile-comparison process. As seen from the test data, lots of users' profiles contain partial, ill-formed or missing (possibly undisclosed) data. In cases when little or no profile information is available, user's friend-list may help to identify a user across domains. We evaluate a number of classification algorithms and present results for Web profile only matching and for extended profile and friend-network matching. We achieve up to 93% success in retrieving identical users with the combined method. This result shows that combining Web profile matching with structure-based matching clearly increases successful classifications.

There is a large space for further research and development in the area of user identification based on their incomplete profiles and there is also opportunity to extend and improve the structure-based identification process.

Acknowledgement

This work is supported in part by Hong Kong Research Grant Council Project No. 619308 and No. 619507.

References

[1] Andrews, N. O., Fox, E. A. (2007). Recent developments in document clustering. *Technical Report* TR-07-35, Computer Science, Virginia Tech.

[2] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S. (2003). *Adaptive name matching in information integration*. IEEE Intelligent Systems, 18 (5) 16–23.

[3] Castells, P., Fernandez, M., Vallet, D. (2007). *An adaptation of the vector-space model for ontology-based information retrieval*. IEEE Trans. on Knowl. and Data Eng., 19 (2) 261–272.

[4] Cohen, W., Ravikumar, P., Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. *In:* Proceedings of IJCAI-03 *Workshop on Information Integration*, p. 73–78.

[5] Dong, X. (2005). Reference reconciliation in complex information spaces. In SIGMOD, p. 85–96.

[6] Dong, X., Halevy, A. (2005). *A platform for personal information management and integration.* In CIDR, p. 119–130.

[7] Dong, X., Halevy, A., Nemes, E., Sigurdsson, S., Domingos, P. (2004). Semex: *Toward on-the-fly personal information integration*. In IIWeb.

[8] Gurney, K. (1997). An introduction to neural networks. CRC Press.

[9] Gusfield, D. (1999). Algorithms on Strings, Trees and Sequences: *Computer Science and Computational Biology*. Cambridge University Press.

[10] Hay, M., Miklau, G., Jensen, D., Towsley, D., Weis, P. (2008). Resisting structural re-identification in anonymized social networks. Proc. VLDB Endow, V. 1, p. 102–114.

[11] Hsu, W. H., Lancaster, J., Paradesi, M. S. R., Weninger, T. (2007). Structural Link Analysis from User Profiles and Friends Networks: *A Feature Construction Approach.* In ICWSM.

[12] Lai, J., Soh, B. (2004). *Similarity score for information filtering thresholds.* In ISCIT, V. 1, p. 216–221.

[13] Li, M., Chen, X., Li, X., Ma, B., Vitanyi, P. M. B. (2004). The similarity metric. *IEEE Transactions on Information Theory*, 50 (12) 3250–3264.

[14] Manning, C., Raghavan, P., Schutze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
[15] Mitchell, T. (1997). *Machine Learning.* McGraw-Hill.
[16] Naderi, H., Rumpler, B. (2007). Three user profile similarity calculation (upsc) methods and their evaluation. In SITIS '07, p. 239–245, Washington, DC, USA.
[17] Naderi, H., Rumpler, B. (2008). Graph-based profile similarity calculation method and evaluation. *In Advances in Information Retrieval*, p. 637–641. Springer Berlin / Heidelberg.
[18] Thiagarajan, R., Manjunath, G., Stumptner, M. (2008). Finding experts by semantic matching of user profiles. *Technical Report HPL*-2008-172, HP Laboratories.
[19] Wee, D. D., Ng, W. K., Sourav, Bhowmick, S. (2002). A survey of web metrics. *ACM Computing Surveys* (34) 469–503.
[20] Xue, Y., Tong, C. S., Zhang, W. (2006). Evaluation of distance measures for nmf-based face recognition. *In Computational Intelligence and Security*, V. 1, p. 651–656.
[21] Zhou, C., Chen, H., Yu, T. (2008). Social network mashup: Ontology-based social network integration for statistic learning. In IRI, p. 143–146.

Author Biographies

Jan Vosecky is a MEng student in the Department of Computer Science at the University of Warwick, United Kingdom. In the academic year 2008/2009, he studied in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). In 2009, he completed a summer research internship at HKUST in the field of social network analysis, supervised by Prof. Vincent Shen.

Dan Hong is currently a postdoctoral research associate in Hong Kong University of Science and Technology supervised by Prof. Vincent Y. Shen. Her research interests are privacy issues in online social networks. She received a Ph.D. in Computer Science and Engineering from the Hong Kong University of Science and Technology in 2009. Before that, she got her Bachelor degree of Engineer from Xi'an Jiaotong University (China) in 2003.

Vincent Y. Shen received his Ph.D. (Electrical Engineering) from Princeton University in 1969. He has taught at the Computer Sciences Department of Purdue University (1969-1985) and worked for the Micro-electronics and Computer Technology Corp. (1985-1990). From 1990 he has been a Professor of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). Professor Shen is a member of the International World Wide Web Conference Steering Committee and a member of ACM's Publications Board. He was a founder of the Hong Kong Mandarin Bible Church and has served as an Elder from 1995.