Inferring User Profiles in Online Social Networks using a Partial Social Graph

Raïssa Yapan Dougnon[1], Philippe Fournier-Viger[1], and Roger Nkambou[2]

Dept. of Computer Science, Université de Moncton, Moncton, Canada
Dept. of Computer Science, Université du quebec à Montréal
{eyd2562,philippe.fournier-viger}@umoncton.ca,nkambou.roger@uqam.ca

Abstract. Most algorithms for user profile inference in online social networks assume that the full social graph is available for training. This assumption is convenient in a research setting. However, in real-life, the full social graph is generally unavailable or may be very costly to obtain or update. Thus, several of these algorithms may be inapplicable or provide poor accuracy. Moreover, current approaches often do not exploit all the rich information that is available in social networks. In this paper, we address these challenges by proposing an algorithm named PGPI (Partial Graph Profile Inference) to accurately infer user profiles under the constraint of a partial social graph and without training. It is to our knowledge, the first algorithm that let the user control the trade-off between the amount of information accessed from the social graph and the accuracy of predictions. Moreover, it is also designed to use rich information about users such as group memberships, views and likes. An experimental evaluation with 11,247 Facebook user profiles shows that PGPI predicts user profiles more accurately and by accessing a smaller part of the social graph than four state-of-the-art algorithms. Moreover, an interesting result is that profile attributes such as status (student/professor) and gender can be predicted with more than 90% accuracy using PGPI.

Key words: social networks, inference, user profiles, partial graph

1 Introduction

In today's society, online social networks have become extremely popular. Various types of social networks are used such as friendship networks (e.g. Facebook and Twitter), professional networks (eg. LinkedIn and ResearchGate) and networks dedicated to specific interests (e.g. Flickr and IMDB). A concern that is often raised about social networks is how to protect users' personal information [7]. To address this concern, much work has been done to help users select how their personal information is shared and with whom, and to propose automatic anonymization techniques [7]. The result is that personal information is better protected but also that less information is now available to companies to provide targeted advertisements and services to their users. To address this issue,

an important sub-field of social network mining is now interested in developing automatic techniques to infer user profiles using the disclosed public information.

Various methods have been proposed to solve this problem such as relational Naïve Bayes classifiers [11], label propagation [8, 10], majority voting [4] and approaches based on linear regression [9], Latent-Dirichlet Allocation [2] and community detection [12]. It was shown that these methods can accurately predict hidden attributes of user profiles in many cases. However, these work suffers from two important limitations. First, the great majority of these approaches assumes the unrealistic assumption that the full social graph is available for training (e.g. label propagation). This is a convenient assumption used by researchers for testing algorithms using large public datasets. However, in real-life, the full social graph is often unavailable (e.g. on Facebook, LinkedIn and Google Plus) or may be very costly to obtain [2]. Moreover, even if the full social graph is available, it may be unpractical to keep it up to date. The result is that several of these methods are inapplicable or provide poor accuracy if the full social graph is not available (see the experiment section of this paper). A few approaches does not assume a full social graph such as majority-voting [4]. However, this latter is limited to only exploring immediate neighbors of a node, and thus do not let the user control the trade-off between the number of nodes accessed and prediction accuracy, which may lead to low accuracy. It is thus an important challenge to develop algorithms that let the user choose the best trade-off between the number of nodes accessed and prediction accuracy.

Second, another important limitation is that several algorithms do not consider the rich information that is generally available on social networks. On one hand, several algorithms consider links between users and user attributes but do not consider other information such as group memberships, "likes" and "views" that are available on social networks such as Facebook and Twitter [1, 4, 6, 8, 10, 11, 12]. On the other hand, several algorithms consider the similarity between user profiles and information such as "likes" but not the links between users and other rich information [2, 9, 13]. But exploiting more information may help to further increase accuracy.

In this paper, we address these challenges. Our contributions are as follows. First, we propose a new lazy algorithm named PGPI (Partial Graph Profile Inference) that can accurately infer user profiles under the constraint of a partial graph and without training. The algorithm is to our knowledge, the first algorithm that let the user select how many nodes of the social graph can be accessed to infer a user profile. This lets the user choose the best trade-off between accuracy versus number of nodes visited.

Second, the algorithm is also designed to be able to use not only information about friendship links and profiles but also group memberships, likes and views, when the information is available. Third, we report results from an extensive empirical evaluation against four state-of-the-art algorithms when applied to 11,247 Facebook user profiles. Results show that the proposed algorithm can provide a considerably higher accuracy while accessing a much smaller number of nodes from the social graph. Moreover, an interesting result is that profile

attributes such as status (student/professor) and gender can be predicted with more than 90% accuracy using PGPI.

The rest of this paper is organized as follows. Section 2, 3, 4, 5 and 6 respectively presents the related work, the problem definition, the proposed algorithm, the experimental evaluation and the conclusion.

2 Related Work

Several work have been done to infer user profiles on online social networks. We briefly review recent work on this topic. Davis Jr et al. [4] inferred location of Twitter users based on relationships to other users with known locations. They used a dataset of about 50,000 users. Their approach consists of performing a majority vote over the locations of directly connected users. An important limitation is that this approach is designed to predict a single attribute based on a single attribute from other user profiles. Having the same goal, Jurgens [8] designed a variation of the popular *label propagation* approach to predict user locations on Twitter/Foursquare social networks. A major limitation is that it is an iterative algorithm that requires the full social graph since it propagates known labels to unlabeled nodes through links between nodes. Li et al. [10] developed an iterative algorithm to deduce LinkedIn user profiles based on relation types. The algorithm is similar to label propagation and requires a large training set to discover relation types.

He et al. [6] proposed an approach consisting of building a Bayesian network based on the full social graph to then predict user attribute values. The approach considers similarity between user profiles and links between users to perform predictions, and was applied to data collected from LiveJournal. Recently, Dong et al. [5] proposed a similar approach based on using graphical-models to predict the age and gender of users. Their study was performed with 1 billion phone and SMS data and 7 millions user profiles. Chaudhari [3] also proposed a graphical model-based approach to infer user profiles, designed to only perform predictions when the probability of making an accurate prediction is high. The approach has shown high accuracy on a dataset of more than 1M Twitter users and a datasets from the Pokac social network. A limitation of these approaches however, it that they all assumes that the full social graph is available for training. Furthermore, they only consider user attributes and links but do not consider additional information such as likes, views and group membership.

A community detection based approach for user profile inference was proposed by Mislove [12]. It was applied on more than 60K Facebook profiles with friendship links. It consists of applying a community detection algorithm and then to infer user profiles based on similarity to other members of the same community. Lindamood et al. [11] also inferred user profiles from Facebook data. They applied a modified Naïve Bayes classifier on 167K Facebook profiles with friendship links, and concluded that if links or attributes are erased, accuracy of the approach can greatly decrease. This raises the challenges of performing accurate predictions using few data. Recently, Blenn et al. [1] utilized bird flocking,

association rule mining and statistical analysis to infer user profiles in a dataset of 3 millions Hyves.nl users. However, all these work assume that the full social graph is available for training and they only use profile information and social links to perform predictions.

Chaabane et al. [2] proposed an approach based on Latent Dirichlet Allocation (LDA) to infer Facebook user profiles. The approach extracts a probabilistic model from music interests and additional information provided from Wikipedia. A drawback is that it requires a large training dataset, which was very difficult and time-consuming to obtain according to the authors [2]. Kosinski et al. [9] also utilized information about user preferences to infer Facebook user profiles. The approach consists of applying Singular Value Decomposition to a large matrix of users/likes and then applying regression to perform prediction. A limitation of this work is that it does not utilize information about links between users and requires a very large training dataset.

Quercia et al. [13] developed an approach to predict the personality of Twitter users. The approach consists of training a decision tree using a large training sets of users tagged with their personality traits, and then use it to predict personality traits of other users. Although this approach was successful, it uses a very limited set of information to perform predictions: number of followers, number of persons followed and list membership count.

3 Problem Definition

As outlined above, most approaches assume a large training set or full social graph. The most common definition of the problem of inferring user profiles is the following [1, 3, 6, 8, 10, 11, 12].

Definition 1 (social graph). A *social graph* $\mathcal{G}$ is a quadruplet $\mathcal{G} = \{N, L, V, A\}$. N is the set of nodes in the social graph. $L \subseteq N \times N$ is a binary relation representing the link (edges) between nodes. Let be m attributes to describe users of the social network such that $V = \{V_1, V_2, ...V_m\}$ contains for each attribute i, the set of possible attribute values V_i. Finally, $A = \{A_1, A_2, ...A_m\}$ contains for each attribute i a relation assigning an attribute value to nodes, that is $A_i \subseteq N \times V_i$.

Example 1. Let be a social graph with three nodes $N = \{John, Alice, Mary\}$ and friendship links $L = \{(John, Mary), (Mary, John), (Mary, Alice), (Alice, Mary)\}$. Consider two attributes *gender* and *status*, respectively called attribute 1 and 2 to describe users. The set of possible attribute values for gender and status are respectively $V_1 = \{male, female\}$ and $V_2 = \{professor, student\}$. The relations assigning attributes values to nodes are $A_1 = \{(John, male), (Alice, female), (Mary, female)\}$ and $A_2 = \{(John, student), (Alice, student), (Mary, professor)\}$.

Definition 2 (Problem of inferring user profiles in a social graph). The problem of inferring the user profile of a node $n \in N$ in a social graph $\mathcal{G}$ is to

correctly guess the attribute values of n using the other information provided in the social graph.

We also consider an extended problem definition where we consider additional information available in social networks such as Facebook (views, likes and group memberships).

Definition 3 (extended social graph). An *extended social graph* $\mathcal{E}$ is a tuple $\mathcal{E} = \{N, L, V, A, G, NG, P, PG, LP, VP\}$ where N, L, V, A are defined as previously. G is a set of groups that a user can be a member of. The relation $NG \subseteq N \times G$ indicates the membership of users to groups. P is a set of publications such as pictures, texts, videos that are posted in groups. PG is a relation $PG \subseteq P \times G$, which associates a publication to the group(s) where it was posted. LP is a relation $LP \subseteq N \times P$, which indicates the publication(s) liked by each user (e.g. the "likes" on Facebook). VP is a relation $VP \subseteq N \times P$, which indicates the publication(s) viewed by each user (e.g. the "views" on Facebook). An observation is that $LP \subseteq VP$.

Example 2. Let be two groups $G = \{book\_club, music\_lovers\}$ such that $NG = \{(John, book\_club), (Mary, book\_club), (Alice, music\_lovers)\}$. Let be two publications $P = \{picture1, picture2\}$ published in the groups $PG = \{(picture1, book\_club), (picture2, music\_lovers)\}$. The publications viewed by users are $VP = \{(John, picture1), (Mary, picture1), (Alice, picture2)\}$ while the publications liked by users are $LP = \{(John, picture1), (Alice, picture2)\}$.

Definition 4 (Problem of inferring user profiles in an extended social graph). The problem of inferring the user profile of a node $n \in N$ in an extended social graph $\mathcal{E}$ is to correctly guess the attribute values of n using the information in the social graph.

But the above definitions assume that the full social graph may be used to perform predictions. In this paper, we define the problem of inferring user profiles using a limited amount of information as follows.

Definition 5 (Problem of inferring user profiles using a partial (extended) social graph). Let $maxFacts \in \mathbf{N}^+$ be a parameter set by the user. The problem of inferring the user profile of a node $n \in N$ using a partial (extended) social graph $\mathcal{E}$ is to accurately predict the attribute values of n by accessing no more than $maxFacts$ facts from the social graph. A *fact* is a node, group or publication from N, G or P (excluding n).

4 The Proposed Algorithm

In this section, we present the proposed PGPI algorithm. We first describe a version PGPI-N that infer user profiles using only nodes and links. Then, we present an alternative version PGPI-G designed for predicting user profiles using only group and publication information (views and likes). Then, we explain how these two versions are combined in the full PGPI algorithm.

4.1 Inferring user profiles using nodes and links

Our proposed algorithm PGPI-N for inferring profiles using nodes and links is inspired by the *label propagation* family of algorithms, which was shown to provide high accuracy [8, 10]. These algorithms suffer however from an important limitation. They are iterative algorithms that require the full social graph for training to propagate attribute values [8].

PGPI-N adapts the idea of label propagation for the case where at most $maxFacts$ facts from the social graph can be accessed to make a prediction. This is possible because PGPI-N is a lazy algorithm (it does not require training), unlike label propagation. To predict an attribute value of a node n, PGPI-N only explores the neighborhood of n, which is restricted by a parameter $maxDistance$.

The PGPI-N algorithm (Algorithm 1) takes as parameter a node n_i, an attribute k to be predicted, the $maxFacts$ and $maxDistance$ parameters and a social graph $\mathcal{G}$. It outputs a predicted value v for attribute k of node n_i. The algorithm first initializes a map M so that it contains a key-value pair $(v, 0)$ for each possible value v for attribute k. The algorithm then performs a breadth-first search. It first initializes a queue Q to store nodes and a set *seen* to remember the already visited nodes, and the node n_i is pushed in the queue.

Then, while the queue is not empty and the number of accessed facts is less than $maxFacts$, the algorithm pops the first node n_j in the queue. Then, the formula $F_{i,j} = W_{i,j}/dist(n_i, n_j)$ is calculated. $W_{i,j}$ and $dist(n_i, n_j)$ are respectively used to weight the influence of node n_j by its similarity and distance to n_i. $W_{i,j}$ is defined as the number of attribute values common to n_i and n_j divided by the number of attributes m. $dist(x, y)$ is the number of edges in the shortest path between n_i and n_j. Then, $F_{i,j}$ is added to the entry in map M for the attribute value of n_j for attribute k. Then, if $dist(x, y) \leq maxDistance$, each node n_h linked to n_j that was not already visited is pushed in the queue and added to the set *seen*. Finally, when the while loop terminates, the attribute value v associated to the largest value in M is returned as the predicted value for n_i. Note that in our implementation, if the $maxFacts$ limit is reached, the algorithm do not perform a prediction to reduce the probability of making an inaccurate prediction.

4.2 Inferring user profiles using groups and publications

PGPI-G is a lazy algorithm designed to predict user attribute values using only group and publication information (views and likes). The algorithm is inspired by majority voting algorithms (e.g. [4]), which have been used for predicting user profiles based on links between users. PGPI-G adapts this idea for groups and publications and also to handle the constraint that at most $maxFacts$ facts from the social graph can be accessed to make a prediction.

The PGPI-G algorithm (Algorithm 2) takes as parameter a node n_i, an attribute k to be predicted, the $maxFacts$ and $maxDistance$ parameters and an extended social graph $\mathcal{E}$. It outputs a predicted value v for attribute k of

Algorithm 1: The PGPI-N algorithm

input : n_i: a node, k: the attribute to be predicted, $maxFacts$ and $maxDistance$: the user-defined thresholds, $\mathcal{G}$: a social graph
output: the predicted attribute value v

1 $M = \{(v, 0) | v \in V_k\}$;
2 Initialize a queue Q;
3 $Q.push(n_i)$;
4 $seen = \{n_i\}$;
5 **while** Q is not empty and $|accessedFacts| < maxFacts$ **do**
6 $\quad n_j = Q.pop()$;
7 $\quad F_{i,j} \leftarrow W_{i,j}/dist(n_i, n_j)$;
8 $\quad$ Add $F_{i,j}$ to the entry of value v in M such that $(n_j, v) \in A_k$;
9 $\quad$ **if** $dist(n_i, n_j) \leq maxDistance$ **then**
10 $\quad\quad$ **foreach** *node* $n_h \neq n_i$ such that $(n_h, n_j) \in L$ and $n_h \notin seen$ **do**
11 $\quad\quad\quad Q.push(n_h)$;
12 $\quad\quad\quad seen \leftarrow seen \cup \{n_h\}$;
13 $\quad\quad$ **end**
14 $\quad$ **end**
15 **end**
16 **return** a value v such that $(v, z) \in M \wedge \nexists (v', z') \in M | z' > z$;

node n_i. The algorithm first initializes a map M so that it contains a key-value pair $(v, 0)$ for each possible value v for attribute k. Then, the algorithm iterates over each member $n_j \neq n_i$ of each group g where n_i is a member, while the number of accessed facts is less than $maxFacts$. For each such node n_j, the formula $Fg_{i,j}$ is calculated to estimate the similarity between n_i and n_j and the similarity between n_i and g. In this formula, $commonLikes(n_i, n_j)$ and $commonViews(n_i, n_j)$ respectively denotes the number of publications liked by both n_i and n_j, while $commonGroups(n_i, n_j)$ is the number of groups common to n_i and n_j. Finally, $commonPopularAttributes(n_i, g)$ is the number of attribute values of n_i that are the same as the most popular attribute values for members of g. Then, $Fg_{i,j}$ is added to the entry in map M for the attribute value of n_j for attribute k. Finally, the attribute value v associated to the largest value in M is returned as the predicted value for n_i. Note that in our implementation, if the $maxFacts$ limit is reached, the algorithm do not perform a prediction to reduce the probability of making an inaccurate prediction.

4.3 Inferring user profiles using nodes, links, groups and publications

The PGPI-N and PGPI-G algorithms have similar design. Both of them are lazy algorithms that update a map M containing key-value pairs and then return the attribute value v associated to the highest value in M as the prediction. Because of this similarity, the algorithms PGPI-N and PGPI-G can be easily combined. We name this combination PGPI. The pseudocode of PGPI is shown in Fig. 3, and is obtained by inserting lines 2 to 15 of PGPI-N before line 8 of PGPI-G.

Algorithm 2: The PGPI-G algorithm

input : n_i: a node, k: the attribute to be predicted, $maxFacts$: a user-defined threshold, $\mathcal{E}$: an extended social graph
output: the predicted attribute value v

$M = \{(v, 0) | v \in V_k\}$;
foreach group $g | (n_i, g) \in NG$ s.t. $|accessedFacts| < maxFacts$ **do**
  foreach person $n_j \neq n_i \in g$ s.t. $|accessedFacts| < maxFacts$ **do**
    $Fg_{i,j} \leftarrow W_{i,j} \times commonLikes(n_i, n_j) \times commonViews(n_i, n_j) \times (commonGroups(n_i, n_j)/|\{(n_i, x) | (n_i, x) \in NG\}|) \times$
    $commonPopularAttributes(n_i, g)$;
    Add $Fg_{i,j}$ to the entry of value v in M such that $(n_j, v) \in A_k$;
  end
end
return a value v such that $(v, z) \in M \wedge \nexists (v', z') \in M | z' > z$;

Moreover, a new parameter named $ratioFacts$ is added. It specifies how much facts of the $maxFacts$ facts can be respectively used by PGPI-N and by PGPI-G to make a prediction. For example, $ratioFacts = 0.3$ means that PGPI-G may use up to 30% of the facts, and thus that PGPI-N may use the other 70%. In our experiment, the best value for this parameter was 0.5.

5 Experimental Evaluation

We performed several experiments to assess the accuracy of the proposed PGPI-N, PGPI-G and PGPI algorithms for predicting attribute values of nodes in a social network. Experiments were performed on a computer with a fourth generation 64 bit Core i5 processor running Windows 8.1 and 8 GB of RAM.

We compared the performance of the proposed algorithms with four state-of-the-art algorithms. The three first are Naïve Bayes classifiers [7]. Naïve Bayes (NB) infer user profiles strictly based on correlation between attribute values. Relational Naïve Bayes (RNB) consider the probability of having friends with specific attribute values. Collective Naïve Bayes (CNB) combines NB and RNB. To be able to compare NB, RNB and CNB with the proposed algorithms, we have adapted them to work with a partial graph. This is simply done by training them with $maxFacts$ users chosen randomly instead of the full social graph. The last algorithm is label propagation (LP) [8]. Because LP requires the full social graph and does not consider the $maxFacts$ parameter, its results are only used as a baseline. For algorithm specific parameters, the best values have been empirically found to provide the best results.

Experiments were carried on a real-life dataset containing 11,247 user profiles collected from the Facebook social network in 2005 [14]. Each user is described according to seven attributes: a student/faculty status flag, gender, major, second major/minor (if applicable), dorm/house, year, and high school. These attributes respectively have 6, 2, 65, 66, 66, 15 and 1,420 possible values. The

Algorithm 3: The PGPI algorithm

input : n_i: a node, k: the attribute to be predicted, $maxFacts$: a user-defined threshold, $\mathcal{E}$: an extended social graph, $ratioFacts$: the ratio of facts to be used by PGPI-G
output: the predicted attribute value v

$M = \{(v, 0) | v \in V_k\}$;
foreach group $g | (n_i, g) \in NG$ s.t. $|accessedFacts| < maxFacts x ratioFacts$ **do**
  foreach person $n_j \neq n_i \in g$ s.t. $|accessedFacts| < maxFacts$ **do**
    $Fg_{i,j} \leftarrow W_{i,j} \times commonLikes(n_i, n_j) \times commThepseudocodeofPGPIisshowninFig.3.onViews(n_i, n_j) \times (commonGroups(n_i, n_j)/|\{(n_i, x) | (n_i, x) \in NG\}|) \times commonPopularAttributes(n_i, g)$;
    Add $Fg_{i,j}$ to the entry of value v in M such that $(n_j, v) \in A_k$;
  end
end
Initialize a queue Q;
$Q.push(n_i)$;
$seen = \{n_i\}$;
while Q is not empty and $|accessedFacts| < maxFacts$ **do**
  $n_j = Q.pop()$;
  $F_{i,j} \leftarrow W_{i,j}/dist(n_i, n_j)$;
  Add $F_{i,j}$ to the entry of value v in M such that $(n_j, v) \in A_k$;
  if $dist(n_i, n_j) \leq maxDistance$ **then**
    foreach *node* $n_h \neq n_i$ such that $(n_h, n_j) \in L$ and $n_h \notin seen$ **do**
      $Q.push(n_h)$;
      $seen \leftarrow seen \cup \{n_h\}$;
    end
  end
end
return a value v such that $(v, z) \in M \wedge \nexists (v', z') \in M | z' > z$;

dataset is a social graph rather than an extended social graph, i.e it does not contain information about group memberships, views and likes. But this information is needed by PGPI-G and PGPI. To address this issue, we have generated synthetic data about group memberships, views and likes. Our data generator takes several parameters as input. To find realistic values for the parameters, we have observed more than 100 real groups on the Facebook network. Parameters are the following. We set the number of groups to 500 and each group to contain between 20 and 250 members, since we observed that most groups are small. Groups are randomly generated. But similar users are more likely to be member of a same group. Furthermore, we limit the number of groups per person to 25. We generated 10,000 publications that are used to represent content shared by users in groups such as pictures, text and videos. A publication may appear in three to five groups. A user has a 50% probability of viewing a publication and a 30% probability of liking a publication that he has viewed. Lastly, a user can

like and view respectively no more than 50 and 100 publications, and a user is more likely to like a publication that users with a similar profile also like.

To compare algorithms, the prediction of an attribute value is said to be a *success* if the prediction is accurate, a *failure* if the prediction is inaccurate or otherwise a *no match* if the algorithm was unable to perform a prediction. Three performance measures are used. The *accuracy* is the number of successful predictions divided by the total number of prediction opportunities. The *coverage* is the number of success or failures divided by the number of prediction opportunities. The *product of accuracy and coverage* (PAC) is the product of the accuracy and coverage and is used to assess the trade-off obtained for these two measures. In the following, the accuracy and coverage are measured as the average for all seven attributes, except when indicated otherwise.

PAC w.r.t number of facts. We have run all algorithms while varying the $maxFacts$ parameter to assess the influence of the number of accessed facts on the product of prediction accuracy and coverage. Fig. 1 shows the overall results. In this figure, the PAC is measured with respect to the number of accessed facts. It can be observed that PGPI provides the best results when 132 to 700 facts are accessed. For less than 132 facts, CNB provides the best results followed by PGPI. PGPI-N provides the second best results for 226 to 306 facts. Note that no results are provided for PGPI-N for more than 306 facts. The reason is that PGPI-N relies solely on links between nodes to perform predictions and the dataset does not contains enough links to let us further increase the number of facts accessed by this algorithm.

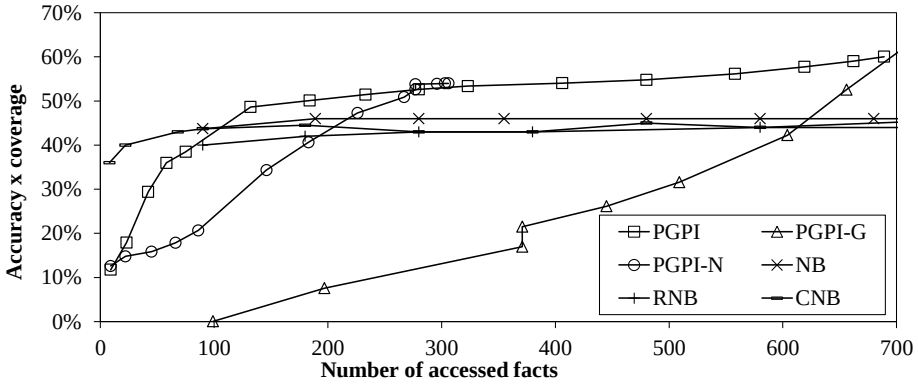

Fig. 1. Product of accuracy and coverage w.r.t. number of accessed facts

The algorithm providing the worst results is LP (not shown in figures). LP provides a maximum accuracy and coverage of respectively 43.2% and 100%. This is not good considering that LP uses the full social graph of more than 10,000 nodes. For the family of Naïve Bayes algorithms, CNB provides the best results until 90 facts are accessed. Then, NB is the best. RNB always provides the worst results among Naïve Bayes algorithms.

Accuracy and coverage w.r.t number of facts. The left and right parts of Fig. 2 separately show the results for accuracy and coverage. In terms of accuracy, it can be observed that PGPI-N, PGPI-G and PGPI always provides much higher accuracy than NB, RNB, CNB and LP (from 9.2% to 36.8% higher accuracy). Thus, the reason why PGPI algorithms have a lower PAC than NB for 132 facts or less is because of coverage and not accuracy. In terms of coverage, the coverage is only shown in Fig. 2 for PGPI-N, PGPI-G and PGPI because the coverage of the other algorithms remains always the same (100% for NB, RNB and CNB, and 94.4% for LP). PGPI has a much higher coverage than PGPI-N and PGPI-G, even when it accesses much less facts. For example, PGPI has a coverage of 87% when it accesses 132 facts, while PGPI-N and PGPI-G have a similar coverage when using respectively 267 and 654 facts. The coverage of PGPI exceeds 95% when using about 400 facts.

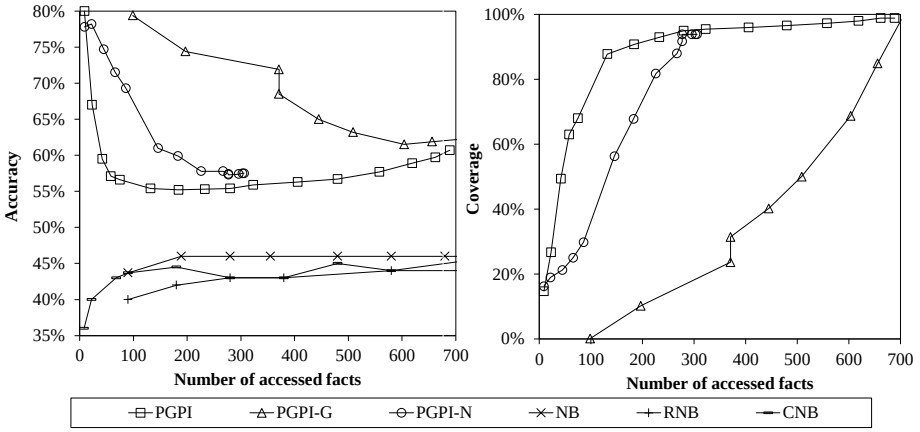

Fig. 2. Accuracy and coverage w.r.t. number of accessed facts

Best results for each attributes. We also analyzed each attributes individually. The best results in terms of accuracy and coverage obtained for each attribute and each algorithm are shown in Table 5. The last column of the table indicates the number of accessed facts to obtain these results.

In terms of accuracy, the best accuracy was achieved by PGPI algorithms for all attributes. For *status*,*gender*, *major*, *minor*, *residence*, *year* and *school*, PGPI algorithms respectively obtained a maximum accuracy of 90.9%, 94.6%, 75.1%, 60.5%, 70.5% and 12.9%. The reason for the low accuracy for *school* is that there are 1,420 possible values and that users are rarely linked to other users with the same value.

The best results in terms of PAC are written in bold. PGPI provides the best PAC for all but one attribute. The second best algorithm is PGPI-N, which provides the second or third best PAC for five of seven attributes. Moreover, it is interesting to see that PGPI-N achieves this result using less than half

the amount of facts used by PGPI. The third best algorithm is PGPI-G, which provides the second or third best PAC for four attributes.

algorithm	status	gender	major	minor	residence	year	school	$\|facts\|$
PGPI acc.	**90.8%**	**87.7%**	**31.5%**	**75.1%**	**59.59%**	**66.86%**	10.3%	684
PGPI cov.	**100%**	**98.9%**	**99%**	**99%**	**99%**	**99%**	99%	684
PGPI-N acc.	89.4%	90.9%	24.5%	73.9%	60.5%	66.77%	**12.9%**	272
PGPI-N cov.	94.4%	94.0%	94.4%	94.4%	94.4%	94.4%	**94.4%**	272
PGPI-G acc.	90.9%	94.6%	35.2%	70.8%	53.0%	70.5%	8.8%	689
PGPI-G cov.	96.6%	72%	72%	72%	72%	72%	72%	689
NB acc.	88.0%	51.1%	16.6%	74.4%	55.6%	26.0%	10.6%	189
NB cov.	100%	100%	100%	100%	100%	100%	100%	189
RNB acc.	80.2%	57.7%	15.0%	74.2%	55.0%	26.2%	9.8%	580
RNB cov.	100%	100%	100%	100%	100%	100%	100%	580
CNB acc.	88.0%	47.3%	9.0%	74.0%	54.8.%	26.0%	10.6%	189
CNB cov.	100%	100%	100%	100%	100%	100%	100%	189
LP acc.	83.0%	50.4%	16.7%	56.3%	49.1%	35.3%	10.1%	10K
LP cov.	94.4%	94.4%	94.4%	94.4%	94.4%	94.4%	94.4%	10K

Table 1. Best accuracy/coverage results for each attribute

Influence of $maxFacts$ and $ratioFacts$ parameters. In the previous experiments, we have analyzed the relationship between the number of accessed facts and the PAC, accuracy and coverage. However, we did not discuss how to choose the $maxFacts$ parameter and how it influences the number of accessed facts for each algorithm.

For Naïve Bayes algorithms and LP, the $maxFacts$ parameter determines the number of nodes to be used for training, and thus directly determines the number of accessed facts.

For PGPI algorithms, we show the influence of $maxFacts$ on the number of accessed facts in Fig. 3. It can be observed that PGPI algorithms generally explore less nodes than $maxFacts$. For PGPI-N, the reason is that some social network users have less than $maxFacts$ friends whithin $maxDistance$ and thus, PGPI-N cannot use more than $maxFacts$ facts to perform a prediction. For PGPI-G, the reason is that some users are members of just a few groups. For PGPI, the number of accessed facts is greater than for PGPI-N but less than for PGPI-G. The reason is that PGPI combines both algorithms and uses the $ratioFacts$ parameter, as previously explained, to decide how much facts can be used by PGPI-N and PGPI-G. The value that we used in previous experiments for this parameter is 0.5, which means to use 50% of the facts for PGPI-G, and thus to use the other half for PGPI-N.

A question that can be raised is how to set the $ratioFacts$ parameter for the PGPI algorithms. In Fig. 4, we show the influence of the $ratioFacts$ parameter on the product of accuracy and coverage. We have measured the PAC w.r.t the number of accessed facts when $ratiofacts$ is respectively set to 0.3, 0.5 and 0.7.

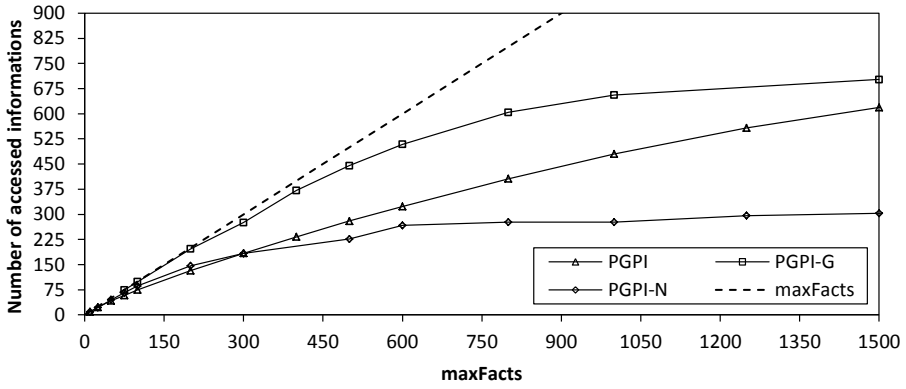

Fig. 3. Number of accessed facts w.r.t. $maxFacts$

Note that setting $ratioFacts$ to 0 and 1 would be respectively equivalent to using PGPI-N and PGPI-G. In Fig. 4, it can be observed that setting $ratioFacts$ to 0.5 allows to obtain the highest PAC. Furthermore, in general a value of 0.5 also gives a high PAC w.r.t the number of accessed facts. A value of 0.3 sometimes gives a higher PAC than 0.5. However, the highest PAC value for 0.3 on overall is lower than that for 0.5. A value of 0.7 almost always gives a lower PAC.

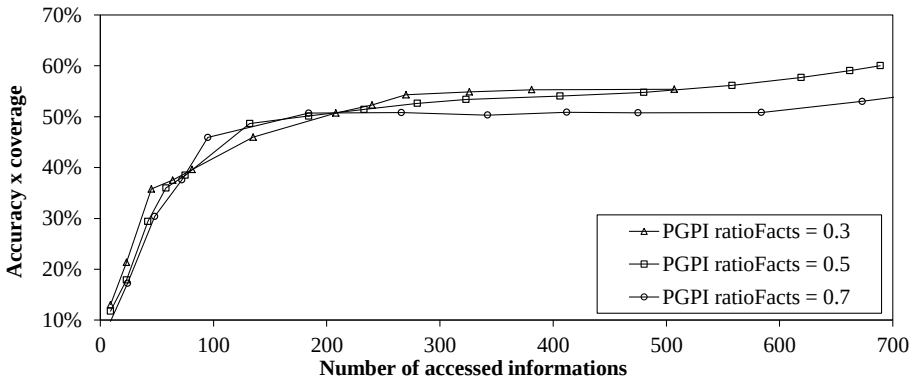

Fig. 4. Product of accuracy and coverage w.r.t. $ratioFacts$

Best results when using the full social graph. We also performed an experiment using the full social graph ($maxFacts = \infty$). The best algorithm in this case is PGPI with an accuracy and coverage of 64.7% and 100%, while the second best algorithm (PGPI-G) achieves 62% and 98.4%.

Influence of the $maxDistance$ parameter. Recall that the PGPI-N and PGPI algorithms utilize a parameter called $maxDistance$. This parameter indicates that two nodes are in the same neighborhood if there are no more than

maxDistance edges in the shortest path between them. In the previous experiments, we have empirically set this parameter to obtain the best results. We now discuss the influence of this parameter on the results of PGPI-N and PGPI.

We have investigated the influence of *maxDistance* on the product of accuracy and coverage, and also on the number of accessed facts. Results are shown in Fig. 5 for PGPI-N. It can be observed that as *maxDistance* is increased, the PAC also increases until it reaches its peak at around $maxDistance = 6$. After that, it remains unchanged. The main reason why it remains unchanged is that the influence of a user on another user is divided by their distance in the PGPI-N algorithm. Thus, farther nodes have a very small influence on each other. Another interesting observation is that as *maxDistance* increases, the number of accessed facts also increases. Thus, if our goal is to use less facts, then *maxDistance* should be set to a small value, whereas if our goal is to achieve the maximum PAC, *maxDistance* should be set to a large value. Lastly, we can also observe that the number of accessed facts does not increase after $maxDistance = 7$. The reason is that the Facebook dataset is quite sparse (i.e. users have few friendship links).

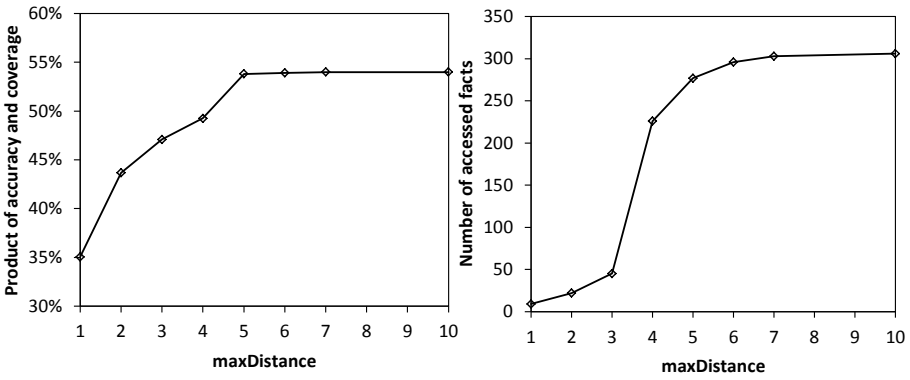

Fig. 5. Product of accuracy and coverage (left) and number of accessed facts (right) w.r.t. *maxDistance* for PGPI-N

Discussion. Overall, PGPI algorithms provide the best accuracy. A key reason is that they are lazy algorithms. They use local facts close to the node to be predicted rather than relying on a general training phase performed beforehand. A second important reason is that more rich information is used by PGPI-G and PGPI (groups, publications). A third reason is that PGPI-N calculates the similarity between users while label propagation does not.

It is interesting to note that although PGPI-G relis on synthetic data in our experiment, PGPI-N does not and performs very well.

6 Conclusion

We have proposed a lazy algorithm named PGPI that can accurately infer user profiles under the constraint of a partial social graph. It is to our knowledge, the first algorithm that let the user control the amount of information accessed from the social graph and the accuracy of predictions. Moreover, the algorithm is designed to use not only node links and profiles as basis to perform predictions but also group memberships, likes and views, if the information is available. An extensive experimental evaluation against four state-of-the-art algorithms shows that PGPI can provide a considerably higher accuracy while accessing a much smaller number of nodes from the social graph to perform predictions. Moreover, an interesting result is that profile attributes such as status (student/professor) and gender can be predicted with more than 90% accuracy using PGPI.

For future work, we plan to further improve the PGPI algorithms by proposing new optimizations and to perform experiments on additional datasets.

References

1. Blenn, N., Doerr, C., Shadravan, N., Van Mieghem, P.: How much do your friends know about you?: reconstructing private information from the friendship graph. In: Proc. of the Fifth Workshop on Social Network Systems, pp. 1–6. ACM (2012)
2. Chaabane, A., Acs, G., Kaafar, M.A.: You are what you like! information leakage through users interests. In: Proc. of the 19th Annual Network and Distributed System Security Symposium, The Internet Society (2012)
3. Chaudhari, G., Avadhanula, V., Sarawagi, S.: A few good predictions: selective node labeling in a social network. In: Proc. of the 7th ACM international conference on Web search and data mining, pp. 353–362. ACM (2014)
4. Davis Jr, C. A., Pappa, G. L., de Oliveira, D. R. R., de L Arcanjo, F.: Inferring the location of twitter messages based on user relationships. Transactions in GIS, 15(6), 735–751 (2011)
5. Dong, Y., Yang, Y., Tang, J., Yang, Y., Chawla, V. N.: Inferring user demographics and social strategies in mobile social networks. In: Proc. of the 20th ACM international conference on Knowledge discovery and data mining, pp. 15–24. ACM (2014)
6. He, J., Chu, W. W., Liu, Z. V.: Inferring privacy information from social networks. In: Proc. of 2006 IEEE International Conference on Intelligence and Security Informatics. pp. 154–165. Springer, Heidelberg (2006)
7. Heatherly, R., Kantarcioglu, M., Thuraisingham, B.: Preventing private information inference attacks on social networks. IEEE Transactions on Knowledge and Data Engineering. 25(8), 1849–1862 (2013)
8. Jurgens, D.: Thats what friends are for: Inferring location in online social media platforms based on social relationships. In: Proc. of the 7th International AAAI Conference on Weblogs and Social Media, pp 273–282, AAAI Press (2013)
9. Kosinski, M., Stillwell, D., Graepel, T.: Private traits and attributes are predictable from digital records of human behavior. National Academy of Sciences, 110(15), 5802–5805 (2013)
10. Li, R., Wang, C., Chang, K. C. C. User profiling in an ego network: co-profiling attributes and relationships. In: Proc. of the 23rd international conference on World wide web, pp. 819–830. ACM (2014)

11. Lindamood, J., Heatherly, R., Kantarcioglu, M., Thuraisingham, B.: Inferring private information using social network data. In: Proc. of the 18th international conference on World wide web, pp. 1145–1146. ACM (2009)
12. Mislove, A., Viswanath, B., Gummadi, K. P., Druschel, P.: You are who you know: inferring user profiles in online social networks. In: Proc. of the 3rd ACM international conference on Web search and data mining, pp. 251–260. ACM (2010)
13. Quercia, D., Kosinski, M., Stillwell, D., Crowcroft, J.: Our Twitter profiles, our selves: Predicting personality with Twitter. In: Proc. of the 3rd IEEE international conference on social computing, pp. 180–185, IEEE Press (2011)
14. Traud, A. L., Mucha, P. J., Porter, M. A.: Social structure of Facebook networks. Physica A: Statistical Mechanics and its Applications. 391(16), 4165–4180 (2012)