Two-phase preference disclosure in attributed social networks

Younes Abid, Abdessamad Imine, Amedeo Napoli, Chedy Raïssi, Michaël Rusinowitch

Two-phase preference disclosure in attributed social networks

Younes Abid, Abdessamad Imine, Amedeo Napoli, Chedy Raïssi and Michaël Rusinowitch

[1] Lorraine University, Cnrs, Inria, 54000 Nancy, France
firstname.lastname@loria.fr

Abstract. In order to demonstrate privacy threats in social networks we show how to infer user preferences by random walks in a multiple graph representing simultaneously attributes and relationships links. For the approach to scale in a first phase we reduce the space of attribute values by partition in balanced homogeneous clusters. Following the Deepwalk approach, the random walks are considered as sentences. Hence unsupervised learning techniques from natural languages processing can be employed in second phase to deduce semantic similarities of some attributes. We conduct initial experiments on real datasets to evaluate our approach.

Keywords: Online Social Network (OSN), Attribute Disclosure Attacks, Privacy

1 Introduction

Social networks offer their users several means to control the visibility of their personal data and publications such as attribute values and friendship links. However even in the case when these policies are properly enforced nowadays data collection techniques and statistical correlations can provide hints on users hidden information [14]. Moreover information leaks from relatives of a user are difficult to control and, for instance, by homophily reasoning [3] an attacker can disclose and exploit sensitive data from a target. Therefore we need to anticipate such disclosure of private information from publicly available data. A way to tackle the problem is to offer users tools that rise their awareness about these privacy breaches. That is we aim to provide people algorithms that try to infer their own hidden attributes, even when social graphs are sparse or friendship links unexploitable, so that they can apply proper countermeasures when such inference are too easy.

In this work, we aim to disclose secret preferences of a social network user for instance, his/her liked movie with high probability of success. Secret preferences are either private or unspecified values of some attribute of the targeted user. The challenge is to predict the secret preference from hundreds of thousands of possible preferences in the network. Therefore to reduce the preference space the first phase consists in clustering attributes values by common likes. For

instance, the values *Star Wars V* and *Star Wars IV* of the attribute movies end up having the same label (i.e. cluster identifier) since they are liked by many common users. By carefully chosing the parameters we end up with relatively homogeneous clusters of balanced sizes. The second phase consists in applying unsupervised learning techniques from natural language processing to disclose the (cluster) label of the secret preference. These techniques have proved to be quite effective for predicting missing links in sparse graphs. Finally, when the label is disclosed we can either further process the cluster content to disclose the secret preference or directly infer the preferences when the clustered values are highly similar, as for instance *Star Wars* episodes.

Let us pinpoint some noticeable features of our approach. Preferences of users for some attribute values are represented by bipartite graphs. Clustering attributes values relies only on users preferences in the considered social network and does not consider external information such as human expertise or information from other websites. We process different graphs of attributes at the same time through random walk to cross latent information about many attributes as detailed in Section 4. For instance, drinks preferences can play a major role to disclose the secretly liked dish of the target. To cope with over-fit problems we assign a weight to each graph in order to quantify its importance in disclosing the secret preference of the targeted user. Weights are parameters validated through off-line tests. We also exploit friendship graphs between users in order to better connect attribute graphs in the random walks.

Related works For space reasons we only discuss closely related works. In [11] the authors propose algorithms to detect whether a sensitive attribute value can be infered from the neighborhood of a target user in a social network. Heatherly et al. [7] seem to be the first that study how to sanitize a social network to prevent inference of social network attributes values. It relies on bayesian classification techniques. The analogous link prediction (recommendation) problem is solved in [1] by exploiting attributes to guide a random walk on graph. The random walk technique has been applied to social representations in [10]. We present here an inference technique that combines attribute clustering and random walks. The method can handle sparse social graphs and attributes with large set of values. The initial clustering allows one to obtain results in a few minutes on large graphs.

2 Social network model

To model social networks for privacy analysis purposes, it is important to take into account their complex structures as well as their rich contents. Infering sensitive and personal information can then be more accurate. We use graphs to model both the structure and the content of social networks. For the network structure, the link-ship networks are modeled either by directed or undirected graphs depending on the social network links type. For instance follow-ship on Twitter are modeled by directed graphs while friendships on Facebook are modeled by undirected graphs. Let $G_l = (U_l, L)$ be the graph of link-ship where U_l is

a set of user nodes and L is a set of links between them. In the same model we use bipartite graphs to represent group membership networks. Let $G_g = (U_g, V_g, P_g)$ be the graph of memberships where U_g is a set of user nodes, V_g is a set of group nodes and P_g is a set of links between user nodes and group nodes.

For modeling the networks contents we use bipartite graphs too. In this work we focus on attributes and omit other contents. Let $G_A = (U_A, V_A, P_A)$ be the graph depicting the preferences of users concerning the attribute A where U_A is a set of user nodes, V_A is a set of nodes representing the different possible values of A and P_A is a set of edges between user nodes U_A and attribute values nodes V_A. P_A represents the preferences (or "likes") of users in U_A for the different attribute values. Figure 1 depicts the detailed model above.

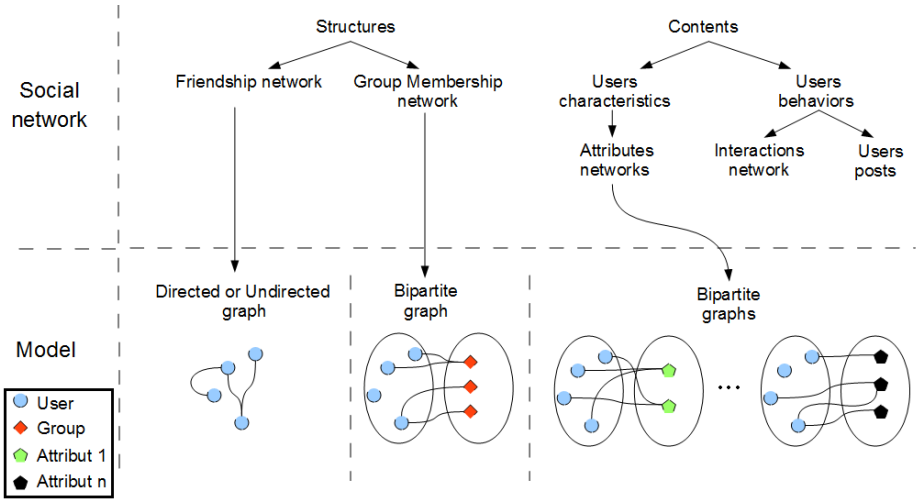

Fig. 1: Social network model for privacy analysis purposes.

Some attributes such as gender have a small set of values. Some others such as music, book and politics have a huge set of possible values. Predicting the favorite book titles or music tunes of a user among scaling thousands of possibilities is hard in a single step. To cope with this problem we decompose the set of possibilities into a few clusters as detailed in Section 3. Our objective will be to predict an attribute value by first predicting the cluster that contains this value. For instance, we aim to predict first the favorite music genres and favorite book genres instead of the favorite music tunes and the favorite book titles directly. Then once such a cluster is determined, infering a prefered item will be easier thanks to the smaller size of clusters compared to the whole set of values. Even a random selection strategy in this last step generates interesting results as shown by our experiments.

3 First phase: clustering targeted attributes values

Since attributes networks are modeled by bipartite graphs, and disclosing the secret preferences will be performed through random walks in the networks, for feasibility of the approach it is important to reduce the number of alternative in paths. For instance, we count 137k community topics, 84k different groups of music and 31k different artists liked by only 15k different users. Therefore, we reduce the space of preferences by clustering attribute values to save computational cost when applying unsupervised learning in Section 5. The problem is alleviated. It consists now of disclosing the secret preferences of the target among a few hundreds of labels instead of tens of thousands of attribute values.

Example. Figure 2 depicts an example of clustering of the attribute movie, $A = movie$. Let $G_{movie} = (U_{movies}, V_{movies}, P_{movies})$ be the bipartite graph relating users to their preferred movies. In this example we aim to partition G_{movie} into $n_l = 2$ subgraphs of almost equally sized disjoints contexts of movies.

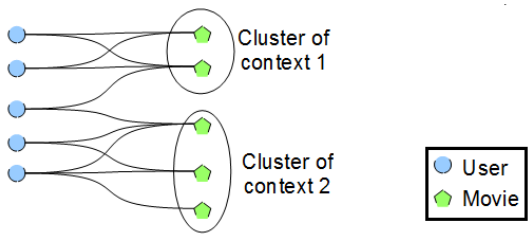

Fig. 2: Example of clustering the values of the attribute movies into disjoint clusters of context.

Clustering all attributes values into contexts requires huge up-to-date knowledge about many fields and many cultures. For instance, Eddie Murphy movies are linked to comedy in 2017 but in 2007 his name was correlated to drama for his role in Dreamgirls for which he picked up his only Oscar nomination. To cope with this problem we cluster the attributes values based on users preferences. However we do not cluster users simultaneously since it is obvious that they can have very different preferences at the same time. For instance, the same user can like both horror and documentary movies. Furthermore, we aim to disclose secret preferences by exploiting information from different graphs, including the friendship graph, as detailed in Section 4.

The problem can be related to a k-way graph partitioning problem since the goal is to divide the set of attribute values into k subset of about equal size. Since we also aim to maximize similarities between attribute values belonging to the same subgraphs, different approaches of dense subgraph discovery could be applied to iteratively seek and cut the densest subgraph from the original graph [8]. However, due to the sparsity of the social graph we consider, the

dense subgraphs are usually small and the algorithms mentioned in [8] end up partitioning the graph into a large number of not equally-sized subgraphs with decreasing densities.

To cope with this issue, we propose a greedy algorithm that adds constraints on the size of subgraphs and the similarity between attribute values of each subgraph. In the following we denote by $|S|$ the cardinal of a set S.

Objective function. Our aimed objective is to find a partition π_l of attribute values in n_l clusters that maximize the similarity between values inside each cluster. We define the similarity between two attribute values v and v' to be the Jaccard coefficient that measures the ratio of their common likes to the union of their likes, where the likes of an attribute value f (say, a movie) is by definition $|\{u \in U_{movies} \text{ s.t.} (u, f) \in P_{movies}\}|$ and denoted by $likes(f)$. That is,

$$similarity(v, v') = \frac{likes(v) \cap likes(v')}{likes(v) \cup likes(v')} \tag{1}$$

For computational efficiency the number of clusters n_l must be small. But if n_l is too small the neural network detailed in Section 5 will be doomed to learn from insufficient data. On the other hand, if n_l is too large the neural network predictions will not be reliable due to over-fitting. Moreover, clusters must be almost equally-sized to avoid fostering a particular label. Therefore we only consider partitions $(c_1, \ldots, c_{n_l})$ of the attribute values satisfying $\sqrt{m} \leq |c_k| \leq 2\sqrt{m}$ for $1 \leq k \leq n_l$, where m is the number of all attribute values, that is the number of movies in our running example. Consequently, the number n_l of clusters satisfies $\frac{\sqrt{m}}{2} \leq n_l \leq \sqrt{m}$. The set of partitions satisfying the constraints above is denoted by Π_l. A good criteria for a candidate cluster c is to maximize the average similarity $similarity(c)$ between all couples of attribute values inside this cluster. Hence the objective function is given by Expression 2.

$$\max_{(c_1,\ldots,c_{n_l}) \in \Pi_l} \frac{1}{n_l} \left(\sum_{k=1}^{n_l} similarity(c_k)\right) \tag{2}$$

Algorithm. Computing the average similarity of a cluster c is expensive due to the quadratic number of couples of values in c. Moreover, the algorithm needs to find the cluster of maximal average similarity among the numerous ones of size between $\sqrt{m}$ and $2\sqrt{m}$. To get around with this problem, we propose a greedy algorithm that computes only the similarity between a cluster of movies and an unlabeled attribute value (that is a value not assigned yet to a cluster). Therefore we define:

$$similarity(c, v) = \frac{\sum_{v' \in c} similarity(v', v)}{|c|} \tag{3}$$

The idea now is to seek, from a set of unlabeled attribute values, an attribute value with maximal similarity with the cluster c (function `seek_max_similar`). Then add the chosen attribute value (`max_similar`) to c. The algorithm keeps adding attribute value to c until it reaches the stop conditions. It then defines next clusters sequentially the same way as detailed in Algorithm 1 until all attribute values are labeled.

Stop conditions. The algorithm stops adding attribute values to the current cluster c when the size of the cluster c is equal to $\texttt{int}(2\sqrt{m})$ or is in $[\sqrt{m}, 2\sqrt{m}-1]$ and one of the two following additional conditions is fulfilled: i) the similarity between c and any of unlabeled attribute values is less than $\frac{1}{2}$; ii) the number of unlabeled attribute values is higher than $\sqrt{m}$. In other word, there exists no sufficiently similar attribute value to add to the current cluster and there is enough unlabeled attribute values to create new clusters. There is also a stopping condition (line 11) when the number of unlabeled attribute values is $\texttt{int}(\sqrt{m})$ to guarantee that the size of the last cluster will be at least $\sqrt{m}$. Finally, the main loop stops when all attribute values are labeled.

Data: $G_A = (U_A, V_A, P_A)$,
Result: π_l ▷ decomposition of V_A into l clusters
1 $B \leftarrow \sqrt{|V_A|}$
2 $V \leftarrow V_A$ ▷ V contains values not assigned to a cluster
3 **while** $|V| > 0$ **do**
4 $\quad c \leftarrow \texttt{one\_most\_liked}(V)$ ▷ initialisation of a new cluster with one element
5 $\quad$ **while** $|c| < 2B$ *and* $|V| > 0$ **do**
6 $\quad\quad$ **if** $B \leq |c|$ **then**
7 $\quad\quad\quad$ **if** $\texttt{max\_similarity}(c, V) < \frac{1}{2}$ `and` $|V| > B$ **then**
8 $\quad\quad\quad\quad$ `break`
9 $\quad\quad\quad$ **end**
10 $\quad\quad\quad$ **if** $|V| = \texttt{int}(B)$ **then**
11 $\quad\quad\quad\quad$ `break`
12 $\quad\quad\quad$ **end**
13 $\quad\quad$ **end**
14 $\quad\quad$ $\texttt{max\_similar} \leftarrow \texttt{seek\_max\_similar}(c, V)$
15 $\quad\quad$ $c \leftarrow c \cup \texttt{max\_similar}$
16 $\quad\quad$ $V \leftarrow V \setminus \texttt{max\_similar}$
17 $\quad$ **end**
18 $\quad$ $\pi_l \leftarrow \pi_l \cup c$
19 **end**

Algorithm 1: Partition of a set of attribute values into clusters.

Size of partitions. We have analyzed the performance of the proposed algorithm with respect to the minimal size of computed clusters, where no cluster can have twice the size of other cluster from the same partition. Tests depicted by Figure 3 show that the choice of the minimal size to be the root square of the size of the set of attribute values yields good results for both very sparse graphs like Users-FastFoods graph (density = 0.0018) and less sparse graphs like the Users-Actors graph (density = 0.012). We note that this choice yields some clusters of high similarity (≥ 0.7), few subgraphs (less than the square root of the number of

attribute values) and relatively high mean similarity compared to all partitions similarities (larger than the mean of the means of all similarities).

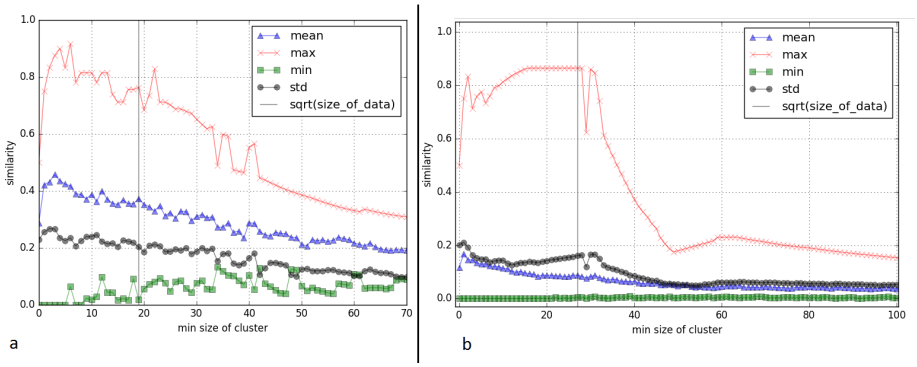

Fig. 3: Variation of partitioned bipartite subgraph similarities with respect to the minimal size of subgraphs, (a) Users-Actors graph: 15k users, 364 actors, (b) Users-FastFoods graph: 15k users, 777 fast foods.

4 Random walks in a social attributed network

In this section we aim to express the latent information in the graphs modeling both the structure and the content of the network into a document that will be processed in Section 5 to disclose secret preferences as detailed in Section 6.

As illustrated in Figure 4, the document is constructed by connecting all graphs through random jumps between them and random walk between their nodes (see also [10]). Since the values of the analyzed attributes are labeled, they are represented by their clusters in the final document. For instance, the first walk depicted by Figure 4 is $[u_1, u_4, v_{2,3}, u_4]$. But for efficiency the walk $[u_1, u_4, c_{2,2}, u_4]$ is stored instead in the document since the value $v_{2,3}$ belongs to the cluster $c_{2,2}$.

Let n be the total number of graphs that model the social network, comprising a link-ship graph $G_1 = G_l = (U_l, L)$ and $n-1$ attribute graphs $G_x = (U_x, V_x, P_x)$. Let U be the set of users in all graphs and n_1 its cardinality. Jumps between two graphs, G_x and G_y, are possible if the current walker state is a user node, say u_z, that belongs to both graphs ($u_z \in U_x \cap U_y$). The walker is allowed to jump from the user node u_z to the graph G_y with a probability $p_{z,y}$. The probability $p_{z,y}$ is defined in Equation 4 where weights are parameters used to quantify the importance of each graph in disclosing secret preferences (e.g., value of some sensitive attribute) of the target.

$$p_{z,y} = \begin{cases} \frac{weight(G_y)}{\sum_{\{1 \le x \le n | u_z \in U_x\}} weight(G_x)} & \text{if } u_z \in U_y \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

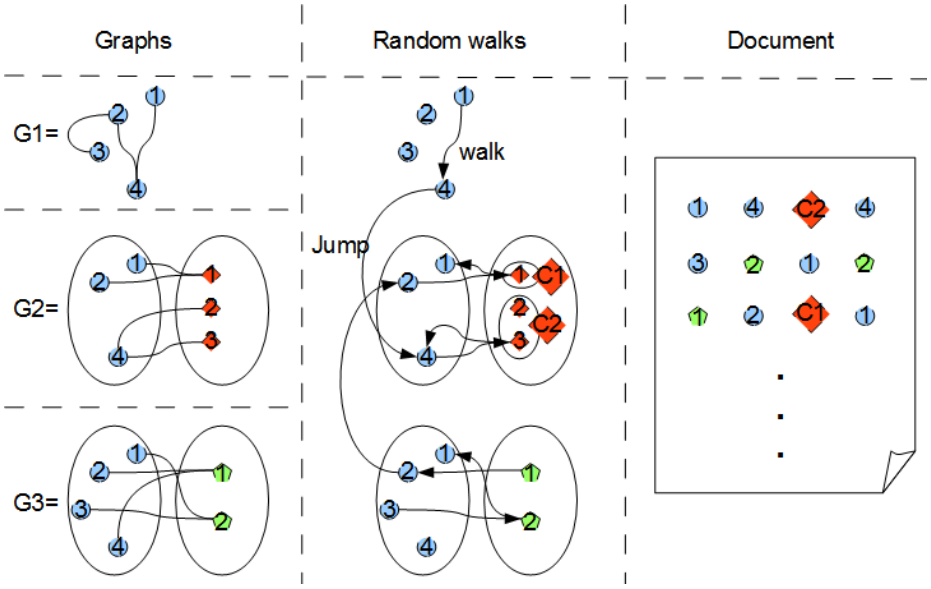

Fig. 4: Example of multi graph random walk.

For each graph $G_y = (U_y, V_y, P_y)$ we define two line stochastic adjacency matrices, $T_{U \times V_y}$ and $T_{V_y \times U}$, and a jump matrix, J_y, that leads to G_y as detailed in 5.

$$J_y = diag(p_{z,y} | u_z \in U)$$

$$T_{U \times V_y}(i,j) = \begin{cases} \frac{1}{deg_y(u_i)} & \text{if}(u_i, v_j) \in P_y \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$T_{V_y \times U}(i,j) = \begin{cases} \frac{1}{deg(v_i)} & \text{if}(u_j, v_i) \in P_y \\ 0 & \text{otherwise} \end{cases}$$

where U is the set of all users in all graphs and $deg_y(u_i)$ is the degree of user u_i in graph G_y.

For the link-ship graph $G_1 = G_l = (U_l, L)$ we define a jump matrix J_1 in the same way as in Equation 5 but only one line stochastic adjacency matrix $T_{U \times U}$ as detailed in Equation 6.

$$T_{U \times U}(i,j) = \begin{cases} \frac{1}{deg_l(u_i)} & \text{if}(u_j, u_i) \in L \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We define now a first order random walk where the next steps probabilities depend only on the current location. Given a source node S we perform a multi-

graph random walk of fixed length l. Steps are generated by the distribution detailed in Expressions7:

$$\forall k \in [2,l], P(s_k|s_{k-1}) = \begin{cases} p_{z,y} \times \frac{1}{deg_y(s_{k-1})} & \text{if } (s_{k-1},s_k) \in P_y \\ & \text{and } s_{k-1} = u_z \text{ and } s_k \in V_y \\ p_{z,l} \times \frac{1}{deg_l(s_{k-1})} & \text{if } (s_{k-1},s_k) \in L \\ & \text{and } s_{k-1} = u_z \text{ and } s_k \in U \\ \frac{1}{deg_y(s_{k-1})} & \text{if } (s_{k-1},s_k) \in P_y \\ & \text{and } s_{k-1} \in V_y \text{ and } s_k \in U \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

The transition matrix is defined by blocks as follows:

$$T = \left[\begin{array}{c|c} J_1 \times T_{U\times U} & J_2 \times T_{U\times V_2} \cdots J_i \times T_{U\times V_i} \cdots J_n \times T_{U\times V_n} \\ \hline T_{V_2\times U} \\ \cdots \\ T_{V_i\times U} \\ \cdots \\ T_{V_n\times U} & 0 \end{array}\right]$$

For the example in Figure 4 the jump matrices and the right stochastic adjacency matrices are as following (assuming $weight(G_1) = weight(G_2) = weight(G_3)$): $J_1 = diag(\frac{1}{3},\frac{1}{3},\frac{1}{2},\frac{1}{3})$, $J_2 = J_3 = diag(\frac{1}{3},\frac{1}{3},0,\frac{1}{3})$ and

$$T_{U\times U} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{bmatrix} \quad T_{U\times V_2} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad T_{U\times V_3} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$T_{V_2\times U} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{V_3\times U} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

Hence, the transition matrix is deduced as following:

$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ v_{2,1} \\ v_{2,2} \\ v_{2,3} \\ v_{3,1} \\ v_{3,2} \end{array} \begin{array}{c} \begin{array}{cccccccc} u_1 & u_2 & u_3 & u_4 & v_{2,1} & v_{2,2} & v_{2,3} & v_{3,1} & v_{3,2} \end{array} \\ \left[\begin{array}{cccc|ccc|cc} 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{6} & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & 0 \\ \hline \frac{1}{2} & \frac{1}{2} & 0 & 0 & & & & & \\ 0 & 0 & 0 & 1 & & & & & \\ 0 & 0 & 0 & 1 & & & 0 & & \\ \hline 0 & \frac{1}{2} & 0 & \frac{1}{2} & & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & & & & & \end{array}\right] \end{array}$$

5 Second phase: applying natural language learning

In Section 4 we performed multi-graph random walk to translate both the structure and the content of the social network into walks. Walks collected in the final document can be interpreted as sentences, where the words are network nodes. Hence, inferring a link between a user node and an attribute value node is similar to the natural languages processing (NLP) problem of estimating the likelihood of words co-occurrence in a corpus.

Here we use a word2vec NLP model [9,5] with skip-gram model and hierarchical Softmax to encode the steps in embeddings. Embeddings where first introduced in 2003 by Bengio et al. [2]. The basic idea is to map one-hot encoded vectors that represent words in a high-dimensional vocabulary space to a continuous vector space with lower dimension. This approach has the virtue of storing the same information in a low-dimensional vector. The skip-gram model for NLP aims to compute words embeddings in order to predict the context of a given word. The input of the neural network is a high-dimensional one-hot vector which represents the target word and its output is a real low-dimensional vector, the *embedding* of the target word, that holds contextual information. The neural network is shallow with one hidden layer and the objective function given by Equation (4) in [10] maximizes the probability of appearance of the target word within a context of w words. This model has the advantage of generating good words representations [9] and it shows good results when it comes to learning structural representations of vertices in a social network [6,10].

Here we adapt this model to the disclosure of secret preferences of users in social networks. To that end, we perform weighted random walks on social graphs representing both friendship structures and attribute preferences of users. In contrast to [10] and [6] where users vertices which have similar friends will be mapped to similar embeddings, in our case both friends and preferences play a role in calibrating embeddings. The relative importance of friends and preferences in computing embeddings are quantified by the graphs weights. With this in mind, profiles that share the most important preferences (of highly weighted graphs) can have similar embeddings even if they do not have similar neighborhood. Moreover, vertices of different types, for instance movies, musics and users, are represented by vectors belonging to the same euclidean space. Hence, secret preferences will be easily predicted through linear algebra as detailed in Section 6. Additionally, by analyzing the variation of accuracy with respect to graph weights we deduce correlations between attributes as detailed in Section 7

6 Ranking attribute values for predicting preferences

Users, clusters of targeted attributes and values of other attributes, are encoded by vectors. The vectors are ranked according to a similarity measure with the target user vector. The inference algorithm will disclose as prefered attribute value one with the smallest rank or highest similarity.

In [12] Schakel et al. show that word2vec unsupervised learning algorithm encodes word semantics by affecting vectors in the same direction for co-occurrent

words during training. Besides, the magnitude of a vector reflects both the frequency of appearance of related words in the corpus and the homogeneity of contexts.

In fact, words that appear in different contexts are represented by vectors that average vectors pointing in different contexts directions. Hence, the final vector magnitude generally decreases with respect to contexts. With that in mind, words used only in few contexts have generally higher magnitude than other words that have the same frequency but are used in more contexts. And the higher the word frequency is, the higher the chance it has to be used in different contexts.

To measure semantic similarity between vertices we apply cosine similarity which is widely used in NLP. This metric measures the cosine of the angle formed by two vectors which represent two different vertices. It yields values in the interval $[-1, 1]$ that quantify the topical similarity between vertices regardless their *centrality*. We discuss why cosine similarity is better adapted than euclidean distance for our purpose. For instance, *Star Wars* and *Titanic* are two famous movies that attract a large audience. The vertices which represent them are connected to many user vertices in the social network. Consequently, their embeddings weight average the embeddings of many dissimilar embeddings of many users. Hence, the embeddings magnitude of these two famous movies are lower than the embeddings magnitude of the other less famous movies. Therefore, the euclidean distance between the vectors which encode *Titanic* and *Star Wars* is lower than the euclidean distance between any of them and the rest of non famous movies. However, the angle between the vectors which represent *Star Wars* and *Titanic* is large due to the fact that they point in different context. With that in mind, if a given user likes Celine Dion song's *My Heart Will Go On*, his encoding vector will points in closer direction to the direction in which points the vector of *Titanic* because the vectors encoding *Titanic* and *My Heart Will Go On* points in similar context. Hence, we can predict that this user might like the *Titanic* movie even if the euclidean distance between the vector which encodes his vertex and the vector which encodes the *Titanic* is large. We note that if a user has few friends and few preferences his vector magnitude will be high. On the other hand, vectors encoding hub users have low magnitude. So their euclidean distances to the vectors encoding *Titanic* and *Star Wars* are low but they do not necessary like them.

7 Experimental results

Dataset. Our dataset contains 15012 Facebook profiles of students and their direct friends. The sample is connected to more than 5 millions Facebook profiles from all over the world. Thus, we take up the challenging task of disclosing secret preferences of users from a highly diverse community with rich background from all over the globe. Our sampled graph of 15012 Facebook users is connected to 1022847 different liked objects. Objects are pages created on Facebook or any other object on the Internet connected to Facebook through Open Graph

protocol (OGp). The OGp is a Facebook invention that enables any web page to become an object in a social graph [1]. Facebook labels objects by types. We counted 1926 different types of object in our sample. Those types of objects are considered as attributes and modeled by bipartite graphs in our model. For instance the most liked type of object in the sample is *community topics* with 137338 different liked objects.

Experimental setup. We detail the example of disclosing the secret travel agency from which the target user books his vacation. We first select target users in the bipartite graph of Users-TravelAgencies and the hide their preferences. The selection algorithm seeks users who like at least λ travel agencies and removes $r\%$ of their preferred ones. We also add a constraint on the travel agencies graph connectivity in order to guarantee that the random walk detailed in Section 4 can reach any travel agency and the neural network detailed in Section 5 can learn about all the travel agencies.

Then we have performed random walks on 7 graphs (6 bipartite graphs for attributes and 1 friendship graph) as in Section 4 and where the travel agencies are labeled (w.r.t. clusters). In this example, we have selected graphs with similar sizes and densities, and various subjects to focus our tests on the subjects rather than the mathematical properties of the graph. Details about the analyzed graphs are given in Table 1. The results of the first clustering phase are also detailed in Table 2.

Graphs	Sizes	Densities
Users-Users	15012 users	8.94×10^{-6}
Users-TravelAgencies	3370 users, 4827 travel agencies	6×10^{-4}
Users-ConsultingAgencies	2288 users, 4176 counsulting agencies	7×10^{-4}
Users-LocalBusiness	2386 users, 4350 local business	5×10^{-4}
Users-Politicians	2554 users, 4589 politicians	9×10^{-4}
Users-AppPages	4396 users, 4244 app pages	8×10^{-4}
Users-Causes	2547 users, 4410 causes	6×10^{-4}

Table 1: Details about the graphs used for learning

Hyper-parameters. We have tuned the hyper-parameters of the neural network as recommended in [10]. That is, the size of the skip-gram window is 10. The length of the walks is 80. The number of repetitions of walks is 10. And the dimension of the embeddings is 128. We rather focus on validating the weights of the different graphs. We used Bayesian optimization as depicted in [13] to automatically tune weights.

Results. We use the area under the ROC curve (AUC) as defined in [4] to measure the accuracy of the infered links. The amount that AUC exceeds 0.5 tells how much the inference algorithm is better than random guessing. The AUC for link prediction problem is computed as following:

[1] `https://developers.facebook.com/docs/sharing/opengraph`

$$\frac{nr_{(nel>esl)} + 0.5 \times nr_{(nel=esl)}}{n_{nel} \times n_{esl}}$$

where n_{nel} is the number of not existing links, n_{esl} is the number of existing but secret links, $nr_{(nel>sl)}$ is the number of couples of a not existing link and a secret link of smaller rank, $nr_{(nel=esl)}$ is the number couples of a not existing link and a secret link of the same rank. Note that AUC value will be 0.5 if the ranks are independent and identically distributed.

In our model links between the targeted user and the travel agencies which belong to the same cluster will have the same rank. Assuming that all clusters have different ranks (69 different cosines coded on 2 bytes in an euclidean space of dimension 128 where vectors are coded on 256 bytes) the AUC can be computed as following:

$$AUC = AUC_1 + AUC_2 \times \frac{nr_{(nel=esl)}}{n_{nel} \times n_{esl}}$$
$$AUC_1 = \frac{nr_{(nel>esl)}}{n_{nel} \times n_{esl}}$$

where AUC_1 is the accuracy of ranking clusters and AUC_2 is the accuracy of ranking values inside the selected cluster c_s (that should contain the secretly prefered value). Due to graph sparsity (only 2 travel agencies are liked by a user in average) we can make the following approximations when the goal is to predict one given secret link at a time ($n_{esl} = 1$).

$$n_{nel} \times n_{sl} \simeq m - 1$$
$$nr_{(nel=esl)} \simeq |c_s| - 1$$
$$AUC \simeq AUC_1 + AUC_2 \times \frac{|c_s|-1}{m-1}$$

Since $m = |TravelAgencies| = 4827$ and $\sqrt{m} - 1 \leq |c_s| - 1 \leq 2\sqrt{m} - 1$ we have

$$0.0146 = \frac{1}{\sqrt{m}+1} = \frac{\sqrt{m}-1}{m-1} \leq \frac{|c_s|-1}{m-1} \leq 2\frac{\sqrt{m}-1}{m-1} = 2\frac{1}{\sqrt{m}+1} = 0.0292$$

For the results depicted in Table 3 the rank inside clusters is generated by independent and identical distribution ($AUC_2 = 0.5$).Therefore $AUC_2 \times \frac{|c_s|-1}{m-1}$ is negligible w.r.t. AUC_1 in that case and does not affect the global accuracy of the prediction. We can observe that the obtained AUC in Table 3 are clearly above 0.5 showing a satisfactory performance from the proposed method. Computation times are in the order of a few minutes. Increasing the number of steps in random walks improves accuracy but affects efficiency.

8 Conclusion

We have proposed a new method for infering hidden attribute values or preferences in social networks. The method relies on first clustering attribute values and then applying efficient machine learning technique from natural language processing. The method has been fully implemented and the first experiments

Full Users-TravelAgencies graph			
Number of travel agencies	4827		
Number of user to travel agency links	9804		
Removed links			
Minimal degree of targets: λ	10		
Percentage of removed links per target: r	10	20	30
Number of targets	69	45	31
Total number of removed links	80	101	106
Graph partitions			
Number of clusters	68	68	68
Maximal number of Travel Agencies in a cluster	138	138	138
Minimal number of Travel Agencies in a cluster	69	69	69
Best cluster similarity between Travel Agencies	0.857	0.614	0.71
Worst cluster similarity between Travel Agencies	0.03	0.003	0.002
Mean of clusters similarities between Travel Agencies	0.12	0.112	0.114
Std of clusters similarity between Travel Agencies	0.14	0.114	0.12

Table 2: Processing of Users-TravelAgencies graph

Graphs	Best weights configurations		
Users-Users	0.2498	0.2154	0.168
Users-TravelAgencies	0.2498	0.138	0.209
Users-CounsultingAgencies	**0.0002**	**0.0002**	0.164
Users-LocalBusiness	**0.0002**	0.2154	**0.111**
Users-Politicians	0.2498	0.2154	0.117
Users-AppPages	**0.0002**	**0.0002**	**0.114**
Users-Causes	0.2498	0.2154	0.117
Percentage of removed links per target: r	10	20	30
Best Mean Accuracy Result (AUC)	0.6836	0.6715	0.6724
	69 targets	45 targets	31 targets

Table 3: AUC and the best weights configurations.

are encouraging. However we need to perform larger scale experiments which is not easy due to the restrictions in crawling social networks and the needs to anonymize properly the collected data. The next step is to develop online tools for users so that they can control privacy leaks from their footprints in social networks.

References

1. L. Backstrom and J. Leskovec. Supervised random walks: Predicting and recommending links in social networks. *CoRR*, abs/1011.4071, 2010.
2. Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
3. I. Elkabani and R. A. A. Khachfeh. Homophily-based link prediction in the facebook online social network: A rough sets approach. *J. Intelligent Systems*, 24(4):491–503, 2015.
4. F. Gao, K. Musial, C. Cooper, and S. Tsoka. Link prediction methods and their accuracy for different social networks and network metrics. *Scientific Programming*, 2015:172879:1–172879:13, 2015.
5. Y. Goldberg and O. Levy. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.
6. A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864, 2016.
7. R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. Preventing private information inference attacks on social networks. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1849–1862, Aug 2013.
8. V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. 2010.
9. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.
10. B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710, 2014.
11. E. Ryu, Y. Rong, J. Li, and A. Machanavajjhala. curso: protect yourself from curse of attribute inference: a social network privacy-analyzer. In *Proceedings of the 3rd ACM SIGMOD Workshop on Databases and Social Networks, DBSocial 2013, New York, NY, USA, June, 23, 2013*, pages 13–18, 2013.
12. A. M. J. Schakel and B. J. Wilson. Measuring word significance using distributed representations of words. *CoRR*, abs/1508.02297, 2015.
13. J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2960–2968, 2012.
14. E. Zheleva, E. Terzi, and L. Getoor. *Privacy in Social Networks*. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, 2012.