A Privacy Recommendation Wizard for Users of Social Networking Sites (Demonstration)

Electrical Engineering & Computer Science, University of Michigan
2260 Hayward Ave. Ann Arbor, MI 48109 USA

ABSTRACT

Privacy is a huge problem for users of social networking sites. While sites like Facebook allow individual users to personalize fine-grained privacy settings, this has proven quite difficult for average users. This demonstration illustrates a machine learning *privacy wizard*, or recommendation tool, that we have built at the University of Michigan. The wizard is based on the underlying observation that real users conceive their privacy preferences (which friends should see which data items) based on an implicit structure. Thus, using a limited amount of carefully-chosen user input, it is usually possible to build a machine learning model that accurately predicts the user's privacy preferences. This model, in turn, can be used to recommend detailed privacy settings for the user. Our demonstration wizard runs as a third-party Facebook application. Conference attendees will be able to "test-drive" the wizard by installing it on their own Facebook accounts.

1. INTRODUCTION

A growing number of social networking sites allow users to customized fine-grained policies controlling access to their personal data. For example, Facebook's "Privacy Settings" page allows users to specify which pieces of profile data (e.g., *Birthdate* or *Religious Views*) each of their friends is allowed to see. Facebook also allows users to create friend *lists*, and then specify whether a piece of profile data is visible or invisible to all friends in a particular list. Unfortunately, studies have shown that it is difficult for users to construct and maintain such policies [3, 6, 8, 12]. On Facebook, for example, a user must manually assign friends to lists. Because the average Facebook user has 130 friends [2], the process can be very time-consuming. Worse, numerous lists may be required for different pieces of data.

Motivated by this problem, we designed and built a machine learning *privacy wizard* for social network data. The goal of the wizard is to automatically recommend detailed privacy settings with minimal effort from the user. The following were some of our main design criteria:

- **Low Effort, High Accuracy:** The wizard may solicit input from the user, but the quantity of input should be limited. A naive approach would ask the user to manually configure her privacy settings for all friends, but this places an undo burden on the user. At the same time, the settings recommended by the wizard should accurately reflect the user's true privacy preferences.
- **Graceful Degradation:** It is difficult to predict the amount of input that a particular user will be willing to provide. As the user provides more input, the accuracy of recommendations should improve. However, it should be possible for the user to quit at any time.
- **Visible Data:** In addition to the user's input, the wizard may also use information that it can gather and process automatically. For confidentiality reasons, however, when assisting a user U, the wizard should only use information that is *already visible* to U. Typically, this includes U's *neighborhood*: U's friends' profiles, and the friend connections among U's friends.

The underlying recommendation algorithms were described and evaluated in an earlier paper [4]. The purpose of this demonstration is to show the tool in action, and to allow conference attendees to try the working system.

2. PRELIMINARIES

2.1 Preferences and Settings

A user's privacy preferences express her willingness to share profile information with each of her friends. Suppose that a particular user has friend set F, and let I denote the set of data items. At the lowest level, the user's privacy preferences can be expressed in terms of a large $|F| \times |I|$ matrix, where each entry is *allow* or *deny*.

We will use the term *privacy preferences* to refer to the user's idealized policy; we will use the term *privacy settings* to refer to the policy that is actually encoded and enforced by the social networking site. The privacy settings can be modeled in exactly the same way. The *setting accuracy* is the proportion of preferences that are correctly encoded by settings.

2.2 Privacy and Community Structure

We conducted a detailed user study, described in [4], which resulted in an important observation: *Real users conceive their privacy preferences according to an implicit structure, or set of rules, and these rules are often related to the community structure of the underlying social network.* This observation is illustrated with an example.

EXAMPLE 1. *Figure 1 shows the* neighborhood network *of a sample user* K*, and her privacy preferences toward Birthdate.*[1] *Each node in the graph represents one of* K*'s friends; there is an edge between two nodes if there is a friend relationship between them.*

[1]User K is one of the authors of this paper. For confidentiality reasons, we do not include raw preference data from other users, but our user study found similar patterns across users [4].

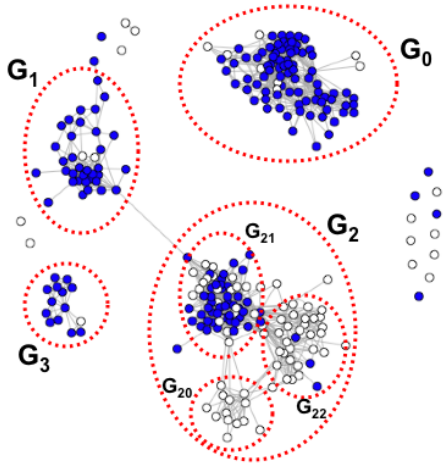

Figure 1: User K's neighborhood graph, and her privacy preferences toward Date of Birth. (Shaded nodes indicate *allow*, and white nodes indicate *deny*.)

In K*'s neighborhood network, observe that there are groups of nodes clustered together. (We plotted Figure 1 using the Fruchterman-Reingold force-based layout, which places topologically near nodes close together, and others far apart.) In social networks research, these groups are commonly called* communities*. We have denoted some communities in the figure: G_0, G_1, etc. Observe also that* K*'s privacy preferences tend to relate to the community structure. She is willing to share her* Date of Birth *with the majority of her friends. However, there are two communities (G_{20} and G_{22}) with whom she does not want to share this data item. This suggests that* K *has implicitly constructed her privacy preferences according to a set of rules related to the community structure of her friend network.*

3. WIZARD OVERVIEW

Based on the observations in the previous section, we have designed and built a unique machine learning *privacy wizard* for social network data [4]. The main components of the wizard are shown in Figure 2.

3.1 User Input

It is widely known that users have difficulty reasoning holistically about privacy and security policies [11, 8], but it is easier to think about simple, concrete examples. Thus, our wizard solicits input by asking the user to assign labels (*allow* or *deny*) for specific friends and data items. Figure 3 shows screenshots of the wizard. A checked box indicates that the user would like to share the data item with the given friend; an unchecked box indicates that she does not want to share the data item with the friend.

3.2 Preference Classifier

A naive approach might ask the user to explicitly label every $(friend, item)$ pair. Of course, this is unreasonable, since the average Facebook user has many friends. Instead, we take the following approach: For each data item i, the wizard learns a *classifier* describing the user's preferences for that item.

Consider a data item i, and let F_L and F_N denote the sets of friends that the user has and has not labeled, respectively. Each of the user's friends f can be described by a *feature vector* (see Section 3.3). Using the labeled examples in F_L, many well-known algorithms can be used to train a classifier. (Our current implementation uses Naive Bayes.) Given an

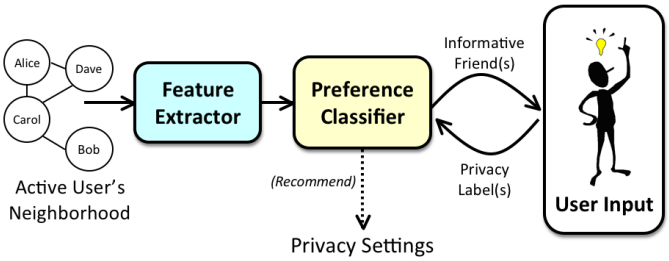

Figure 2: Wizard Component Overview

unlabeled feature vector (friend), the trained classifier can be used to predict that friend's label. Thus, the wizard asks the user to label a small fraction of friends, trains a classifier, and uses the classifier to recommend settings for the remaining friends in F_N.

3.3 Feature Extraction

In order to build a good classifier, it is important to choose a good set of features. In our earlier work, we found that it is very effective to extract *communities* from the user's neighborhood network (i.e., the user's friends, and the friend connections between them) [4].

In social network research, a network is often said to have a *community* structure if its nodes can naturally be separated into groups, where the nodes in each group are densely connected, but there are few connections between disparate groups. For example, in Figure 1, it is easy to see several such communities, some of which we have circled and labeled. From a sociological perspective, two people in the same community are more likely to know one another than two people who are not in the same community.

Numerous algorithms have been developed for finding communities [5]. In our current implementation, we use a hierarchical variation of Newman's fast-greedy modularity optimization [9]. At a high level, the algorithm works as follows: (1) First, partition the full network into communities using the fast-greedy algorithm and maximizing *modularity* [10], (2) For each resulting community, discard the surrounding network, and view the community as its own network, (3) Repeat this process recursively until each community contains a single node.

Notice that the resulting communities retain some hierarchical structure (i.e., larger communities that fully contain several smaller communities). For example, in Figure 1, we have marked a total of seven communities, but community G_2 fully contains three smaller communities. This is useful because a user's privacy preferences can be expressed at varying degrees of granularity.

Finally, each extracted community can be regarded as a boolean feature (i.e., a particular friend belongs to the community or not).

EXAMPLE 2. *As a simple example, Figure 4 shows a set of labeled friends, using a feature-vector representation with seven community features. For example, Bob is a member of the extracted communities G_2 and G_{20}. The user has assigned labels to Alice and Bob for data item Birthdate, but Carol's label is still unknown.*

3.4 Informative Input via Active Learning

In order to learn the preference classifier, the wizard needs to get some input from the user, as described above. Since the user's attention is limited, it is critical for the wizard to "ask the right questions," or select the most informative

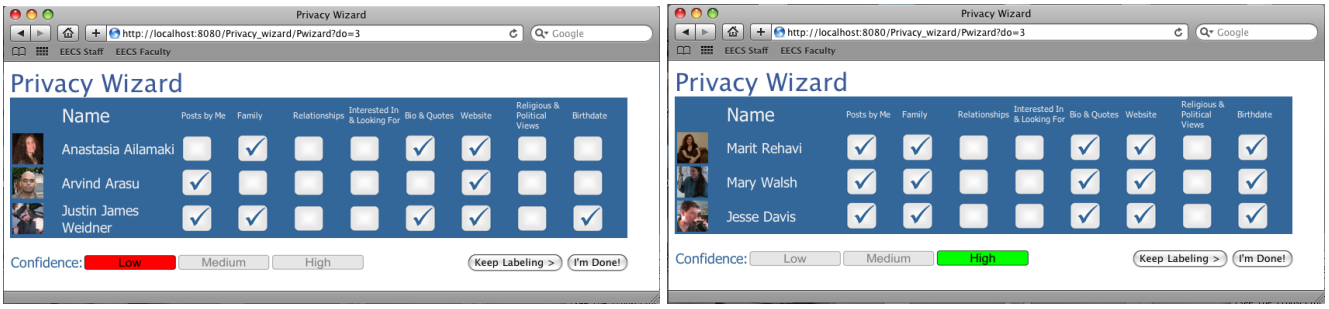

Figure 3: Privacy wizard screenshots. The user provides input by labeling carefully-chosen friends. The confidence indicator bar on the bottom indicates when it is OK to stop labeling friends.

	G_0	G_1	G_2	G_{20}	G_{21}	G_{22}	G_3	Pref. Label (Birthdate)
(Alice Adams)	0	1	0	0	0	0	0	*allow*
(Bob Baker)	0	0	1	1	0	0	0	*deny*
(Carol Cooper)	1	0	0	0	0	0	0	?

Figure 4: Example friend data with extracted features, including community-based features (G_0, G_1, etc.)

friends for the user to label. To explain this intuitively, if the wizard is already extremely confident, based on past input, that the user does not want to share her Political Views with friends in community G_2, then it does not make sense to continue asking the user to label friends from that community.

In our current implementation, we solicit informative input using an active learning paradigm known as *uncertainty sampling* [7]. Essentially, uncertainty sampling operates in rounds. During each round, for each unlabeled example, the classifier predicts the probability distribution over class labels ($p(allow)$ and $p(deny)$), as well as an *uncertainty score*. In our current implementation, the uncertainty score is the entropy of the predicted distribution. For example, if the classifier predicts with probability 1.0 that the label associated with a particular example is *allow*, then the entropy is 0.0, indicating low uncertainty. In contrast, if the predicted distribution is even (i.e., $p(allow) = 0.5$ and $p(deny) = 0.5$), then the entropy is 1.0. The wizard displays k of the *least certain* friends for the user to label; the display is automatically populated with the most likely labels as the default options.[2]

This approach works well because it allows the wizard to learn a good model from a limited quantity of labeled training data. (In our earlier work, we found that an average user could obtain over 90% accuracy by labeling only 10% of her friends [4].) Also, while the user can improve the setting accuracy by labeling more friends, the wizard adapts gracefully if the user quits at any time.

3.5 Progress Indicators

Finally, although the user can quit labeling at any time, it is still important to provide some cues about the quality of the current model and predictions. For this purpose, we include a *confidence indicator* bar, which is shown at the bottom of the screenshots in Figure 3. The current confidence level (high, medium, or low) is chosen based on the single (friend, data item) pair with the highest uncertainty score (highest entropy).

[2]Strictly speaking, the wizard does not display the k least certain friends. It also incorporates a diversity criterion to ensure that the k displayed friends are not too similar to one another.

4. DEMONSTRATION SCENARIO AND REQUIREMENTS

We have implemented the wizard using the Facebook open-development platform [1]. Conference attendees will be able to test-drive the wizard by installing the application on their own Facebook accounts. In order to run the demonstration, we will need reliable Internet connectivity.

After running the wizard, we provide the user with two main results: First, the recommended privacy settings are displayed on the screen. Second, the system generates a Javascript file; if the user chooses to run this code, her Facebook privacy settings will actually be updated according to the recommendation.

5. REFERENCES

[1] Facebook development platform. http://developers.facebook.com/.
[2] Facebook statistics. http://www.facebook.com/press/info.php?statistics.
[3] A. Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *PET*, 2006.
[4] L. Fang and K. LeFevre. Privacy wizards for social networking sites. WWW, 2010.
[5] S. Fortunato. Community detection in graphs. http://arxiv.org/abs/0906.0612v1 (Preprint), 2009.
[6] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *WPES*, 2005.
[7] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *SIGIR*, 1994.
[8] H. Lipford, A. Besmer, and J. Watson. Understanding privacy settings in facebook with an audience view. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*, 2008.
[9] M. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), 2004.
[10] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.
[11] R. Reeder, L. Bauer, L. Cranor, M. Reiter, K. Bacon, K. How, and H. Strong. Expandable grides for visualizing and authoring computer security policies. In *CHI*, 2008.
[12] K. Strater and H. Lipford. Strategies and struggles with privacy in an online social networking community. In *British Computer Society Conference on Human-Computer Interaction*, 2008.