Privacy Wizards for Social Networking Sites

Lujun Fang and Kristen LeFevre
Electrical Engineering & Computer Science, University of Michigan
2260 Hayward Ave. Ann Arbor, MI 48109 USA
ljfang@umich.edu,klefevre@umich.edu

ABSTRACT

Privacy is an enormous problem in online social networking sites. While sites such as Facebook allow users fine-grained control over who can see their profiles, it is difficult for average users to specify this kind of detailed policy.

In this paper, we propose a template for the design of a social networking *privacy wizard*. The intuition for the design comes from the observation that real users conceive their privacy preferences (which friends should be able to see which information) based on an implicit set of rules. Thus, with a limited amount of user input, it is usually possible to build a machine learning model that concisely describes a particular user's preferences, and then use this model to configure the user's privacy settings automatically.

As an instance of this general framework, we have built a wizard based on an active learning paradigm called *uncertainty sampling*. The wizard iteratively asks the user to assign privacy "labels" to selected ("informative") friends, and it uses this input to construct a classifier, which can in turn be used to automatically assign privileges to the rest of the user's (unlabeled) friends.

To evaluate our approach, we collected detailed privacy preference data from 45 real Facebook users. Our study revealed two important things. First, real users tend to conceive their privacy preferences in terms of *communities*, which can easily be extracted from a social network graph using existing techniques. Second, our active learning wizard, using communities as features, is able to recommend high-accuracy privacy settings using less user input than existing policy-specification tools.

Categories and Subject Descriptors

H.2.7 [**Information Systems**]: Security, integrity, and protection

General Terms

Security

Keywords

Social Network Privacy, Usability, Active Learning

1. INTRODUCTION

Social networking sites (e.g., Facebook, MySpace, Friendster, Orkut, etc.) are websites that enable people to share information and communicate with friends online. At the same time, users typically do not want to share all of their information with everyone, and privacy has emerged as a serious concern.

A growing number of social networking and social media sites allow users to customize their own privacy policies. For example, Facebook has a "Privacy Settings" page, which allows users to specify which pieces of profile data each friend is allowed to view. 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 consistently shown that users struggle to express and maintain such policies [4, 13, 22, 27, 39], due in part to complex and unusable interfaces [39]. On Facebook, for example, the 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 since a user's privacy preferences can be different for different pieces of profile data (e.g., *Home Address* vs. *Religious Views*).

Clearly, there is a need for something better. In this paper, we propose the first *privacy wizard* for social networking sites. The goal of the wizard is to automatically configure a user's privacy settings with minimal effort from the user.

1.1 Challenges

The goal of a *privacy wizard* is to automatically configure a user's privacy settings using only a small amount of effort from the user. The design and implementation of a suitable wizard present a number of difficult challenges. Ideally, the wizard should satisfy the following requirements:

- **Low Effort, High Accuracy:** The wizard may solicit input from the user. Research has shown, however, that users have trouble reasoning holistically about privacy and security policies [35, 27]. Thus, the user's input should be simple in form, and also limited in quantity.

  At the same time, the settings chosen by the wizard should accurately reflect the user's true privacy preferences. A naive approach would ask the user to manually configure her privacy settings for all friends. While this approach may produce perfect accuracy if carried to completion, it also places an undo burden on the user.
- **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 the resulting settings should improve. However, the wizad should assume that the user can 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

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

automatically (i.e., without any user intervention). 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*: the information visible to U in U's friends' profiles, and the friend connections among U's friends.

- **Incrementality:** The settings constructed by the wizard should gracefully evolve as the user adds new friends.

1.2 Summary of Contributions

In response to these challenges, we developed a generic framework for the design of a privacy wizard, which is described in Section 2. One of the key insights behind our approach is the observation that real users conceive their privacy preferences according to an implicit set of rules. Thus, using machine learning techniques, and limited user input, it is possible to infer a *privacy-preference model* (i.e., a compact representation of the rules by which an individual conceives her privacy preferences). This model, in turn, can be used to configure the user's settings automatically.

As one instance of the generic approach, we have developed the active-learning privacy wizard described in Section 3. The wizard implements the privacy-preference model by learning a classifier. In the classifier, the features used to describe each friend, including community membership, are extracted automatically from the data visible to the user. The wizard provides very simple user interactions: Leveraging the machine learning paradigm of *active learning*, it iteratively asks the user to assign privacy *labels* (e.g., *allow* or *deny*) to specific, carefully-selected, friends. As the user provides more input, the quality of the classifier improves, but the user can stop at any time. Further, the wizard adapts gracefully as the user adds new friends.

The basic wizard is extremely simple to use, and well-suited for typical (non-technical) users. However, advanced technical users may complain that it does not allow them to view or directly manipulate the resulting privacy-preference model. Thus, in Section 4 we describe a set of visualization and modification tools for advanced users.

To evaluate our solution, we conducted a detailed study of real users. Using raw privacy preferences, which we collected from 45 real Facebook users, the experiments in Section 5 show two important things: First, our wizard achieves a significantly better effort-accuracy tradeoff than alternative policy-specification tools. On average, if a user labels just 25 (of over 200) friends, the wizard configures the user's settings with $> 90\%$ accuracy. Second, *communities* extracted from a user's neighborood are extremely useful for predicting privacy preferences.

2. WIZARD OVERVIEW

2.1 Preliminaries

A user's privacy preferences express her willingness (or unwillingness) to share profile information with each of her friends. Formally, for a particular user, we will denote the user's set of friends as F. We will denote the set of information items in the user's profile as I. At the lowest level, the user's privacy preferences can be expressed in terms of the function $pref : I \times F \rightarrow \{allow, deny\}$. If $pref(i, f) = allow$, this means that it is the user's preference to allow friend f to see profile item i.

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 also be viewed as a function: $setting : I \times F \rightarrow \{allow, deny\}$.

For a particular friend set F and data item set I, the *setting accuracy* is the proportion of preferences correctly encoded by settings.[1]

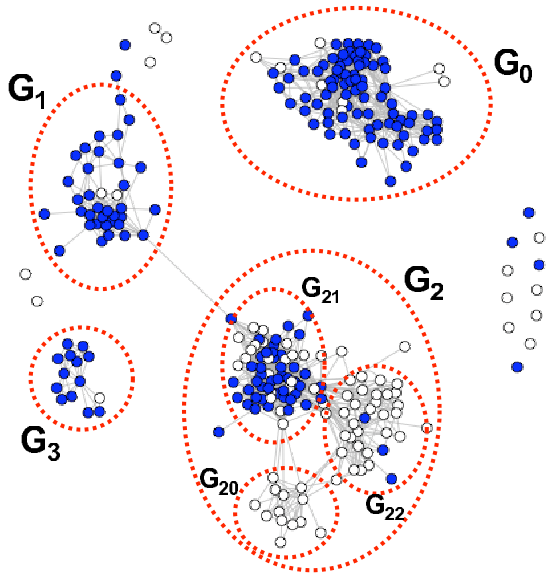

Figure 1: User K's neighborhood graph, and her privacy preferences toward Date of Birth. (Shaded nodes indicate *allow*, and white nodes indicate *deny*.) Notice that K's privacy preferences are highly correlated with the *community* structure of the graph.

2.2 Generic Wizard Design

This section describes the design of a generic *privacy wizard*. Motivating the design is the fundamental observation that real social network users actually conceive their privacy preferences based on unique sets of implicit rules. The details of our user study are postponed to Section 5.1, but the intuition is illustrated with an example.

EXAMPLE 1. *Figure 1 shows the neighborhood of a sample user* K*, and her privacy preferences toward Date of Birth.*[2] *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.*

In User K*'s neighborhood network, observe that there are group 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 manually denoted some apparent communities on the figure:* G_0, G_1, *etc. Observe also that User* K*'s privacy preferences tend to break down along the lines of the community structure. She is willing to share her* Date of Birth *with the majority of her friends. However, there are two communities (labeled* G_{20}

[1]Formally, $Accuracy = \frac{|\{(i,f) \in I \times F : pref(i,f) = setting(i,f)\}|}{|I \times F|}$.

[2]User K is the second author of this paper. Her preferences are included as an illustrative example. To protect confidentiality, we do not include the raw preference data collected from actual study subjects.

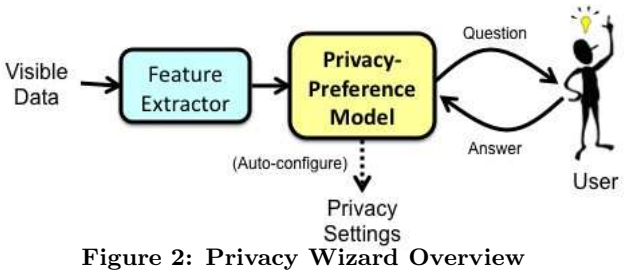

Figure 2: Privacy Wizard Overview

and G_{22}) with whom she does not want to share this data item. This suggests that User K *has implicitly constructed her privacy preferences according to a set of rules, and that these rules are related to the underlying community structure of her friend network.*

Based on this observation, and in response to the requirements outlined in the introduction, we propose a generic framework for constructing a privacy wizard, which is shown in Figure 2. The framework consists of three main parts:

- **User Input:** The wizard solicits input from the user regarding her privacy preferences. In the most general case, this is in the form of questions and answers. At any point, the user may quit answering questions.
- **Feature Extraction:** Using the information visible to the user, the wizard selects a feature space $\vec{X}$. Each of the user's friends can be described using a feature vector $\vec{x}$ in this space.
- **Privacy-Preference Model:** Using the extracted features and user input, the privacy wizard constructs a *privacy-preference model*, which is some inferred characterization of the rules by which the user conceives her privacy preferences. This model is used to automatically configure the user's privacy settings. As the user provides more input, or adds new friends, the privacy-preference model and configured settings should adapt automatically.

Of course, each of these components is quite general. In the next section, we will describe one specific instantiation of the framework.

3. ACTIVE LEARNING WIZARD

In this section, we will describe a specific instantiation of the generic framework outlined in the previous section. In building the wizard, one of our goals was to keep the user interaction as simple as possible. It is widely accepted that users have difficulty reasoning holistically about privacy and security policies [35, 27]. In contrast, it is easier to reason about simple, concrete examples. Thus, our privacy wizard solicits input from the user by asking her preference (*allow* or *deny*) for specific *(data item, friend)* pairs $(i, f) \in I \times F$. Without loss of generality, in the remainder of this section, we will assume that the data item i is fixed (e.g., Date of Birth), and the wizard simply asks the user to assign a preference *label* to a selected friend $f \in F$.

EXAMPLE 2. *The privacy wizard interacts with the user by asking a series of simple questions. For example:*

Would you like to share DATE OF BIRTH with ...
Alice Adams? (y/n)
Bob Baker? (y/n)
Carol Cooper? (y/n) ...

Given this form of user interaction, it is natural to view the preference model as a binary *classifier*, trained using the friends that the user has labeled. However, because the user's effort is limited and unpredictable, it is important that the privacy wizard "ask the right questions," or intelligently request that the user provide labels to the most informative unlabeled friends. In the machine learning literature, this scenario, in which the learner can actively query the user for labels, is commonly known as *active learning*.

In the remainder of this section, we will first describe the construction of a classifier for predicting privacy preferences. Then, we will describe feature extraction, based on visible data, including automatically-extracted communities. Finally, we will describe the application of a particular active learning technique known as *uncertainty sampling* [26].

3.1 Preference Model as a Classifier

For a particular social network user, it is natural to view the privacy-preference model as a classifier. Each of the user's friends f can be represented by a vector of extracted features $\vec{x}$ in a feature space $\vec{X}$ (see Section 3.2).

Using a set of labeled training examples (in this case, labeled friends) $F_{labeled}$, many well-known algorithms (e.g., Decision Trees, Naive Bayes, Nearest Neighbor, etc.) can be used to infer a *classifier*. (We tried several such algorithms in our experiments.) In the most general sense, the classifier uses a feature vector representation of a friend to predict the friend's privacy label. Formally, for a particular data item $i \in I$, the classifier can be viewed as a function of the form $\widehat{pref} : \vec{X} \rightarrow \{allow, deny\}$

The resulting classifier can be used to predict the user's privacy preferences for unlabeled friends in $F_{unlabeled}$. It is important to point out that, in the context of the privacy wizard, we will assume that the labels the user assigns explicitly to friends in $F_{labeled}$ are always correct. The classifier $\widehat{pref}$ is only used to configure the user's privacy settings for friends whom she has not labeled explicitly.

3.2 Feature Extraction

In order to build a reasonable classifier, it is important to select a good set of features. For the purposes of this work, we considered two main types of features: features based on extracted *communities*, and other profile data.

- **Community Structure:** Let $F_{labeled}$ and $F_{unlabeled}$ denote the user's labeled and unlabeled friends, respectively. We can automatically extract a set of *communities* from the user's full neighborhood (i.e., $F_{labeled} \cup F_{unlabeled}$, and the edges connecting these friends) using techniques described in Section 3.2.1. Each extracted community can be regarded as a boolean feature (i.e., a particular friend belongs to the community or not). For example, suppose that we have extracted a community G_1 from the network. If a particular friend belongs to G_1, then that friend has feature value $G_1 = 1$; otherwise, $G_1 = 0$.
- **Other Profile Information:** There are additional attributes in the user's friends' profiles that can be used as features. Since our study wizard is implemented in the context of Facebook, we consider the following when they are visible to the user: Gender, Age, Education history (high school and college), Work History, Relationship Status, Political Views, and Religious Views. These items can be directly translated to features. For example,

	Age	Gender	G_0	G_1	G_2	G_{20}	G_{21}	G_{22}	G_3	Obama_Fan	Pref. Label (Date of Birth)
(Alice Adams)	25	F	0	1	0	0	0	0	0	1	*allow*
(Bob Baker)	18	M	0	0	1	1	0	0	0	0	*deny*
(Carol Cooper)	30	F	1	0	0	0	0	0	0	0	?

Figure 3: Example friend data with extracted features, including community-based features ($G_0, G_1,$ etc.)

Gender has nominal values $\{male, female\}$. In addition, the user's friends' online activities can be used, including Facebook groups, "fan" pages, events, and tagged photos. For these, we use binary features, which indicate whether a particular friend is a member.

EXAMPLE 3. *As a simple example, Figure 3 shows a set of labeled friends, using a feature-vector representation. For example, Bob is a member of the extracted communities G_2 and G_{20}, and Alice is a "fan" of Barack Obama. The user has assigned preference labels to Alice and Bob, but Carol's label is unknown.*

In the remainder of this section, we briefly describe how we extract communities from the user's neighborhood network.

3.2.1 Community-Based Features

In the study of social networks, 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 individuals in the same community are relatively more likely to know one another than two individuals who are not in the same community.

Numerous algorithms have been developed for finding communities. (For an extensive survey on the topic, please see [18].) In this paper, our primary goal is not to develop new community-finding algorithms. Instead, we will simply apply a common algorithm based on the idea of *edge betweenness* [33]. Please note that, in all cases, this algorithm can be replaced with any hierarchical (agglomerative or divisive) community-finding algorithm.

When finding communities in a social network, it is often difficult to know the right number of communities ahead of time. For example, in Figure 1, G_0, G_1, and G_3 seem to be well-defined communities. Looking at G_2, however, it is not immediately clear whether this is a single community, or if it makes sense to further divide it into sub-communities G_{20}, G_{21}, and G_{22}. This problem can be addressed in several different ways. One option is to partition the network into communities to maximize the *modularity* score [33]. In this case, the number of communities is automatically selected based on modularity.

For the purposes of this work, it is not necessary to partition the graph into a single set of communities. Because a user's privacy preferences can be expressed at varying degrees of granularity, it makes sense to 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.

In the remainder of the paper, we will extract multi-granularity communities according to the following process: (1) First, we partition the full network into communities using the edge-betweenness algorithm and maximizing modularity. (2) For each resulting community, we discard the surrounding network, and view the community as its own network. (3) We repeat this process recursively until each community contains a single node.

Observe the community structure is only re-calculated when new friends are added. Typically, this will be done offline. For the neighborhood networks typically encountered in online social networks, which contain on the order of several hundred friends, we do not expect the performance of the community-finding algorithm to be a major issue.

3.3 Uncertainty Sampling

Ultimately, the accuracy achieved by the wizard depends on two factors: (1) The number of friends that the user labels explicitly (these are always assumed to be correct), and (2) The accuracy of the inferred classifier $\widehat{pref}$ in predicting the labels of unlabeled friends. Since the amount of effort a user is willing to devote to labeling friends is limited and unpredictable, it is important that we be able to learn an accurate classifier with a limited amount of training data.

Motivated by the graceful degradation principle, which aims to achieve the best accuracy possible, with the understanding that the user may quit labeling friends at any time, we have chosen to address this problem using an active learning paradigm known as *uncertainty sampling* [26].

Uncertainty sampling consists of two phases:

1. In the *sampling phase*, the wizard selects friends for the user to label.

2. Then, during the *classifier construction phase*, the wizard uses the labeled examples to build the actual classifier ($\widehat{pref}$), which is used to configure the user's settings.

The *sampling phase* works as follows. Initially, all of a user's friends are unlabeled. The sampling proceeds in rounds. During each round, the wizard selects the k unlabeled friends about which it is most *uncertain*, and asks the user to assign labels to these friends. The process terminates after all friends have been explicitly labeled, or when the user abandons the process, whichever comes first.[3]

The *uncertainty* of a class label is traditionally measured by training a classifier (using labeled training data $F_{labeled}$), and using this classifier to predict the distribution of class labels associated with each friend in $F_{unlabeled}$. In our case, there are two possible class labels, and the predicted distribution of class labels is of the form $P(allow) = P_{allow}$, $P(deny) = P_{deny}$, where $P_{allow} \in [0, 1.0]$, $P_{deny} \in [0, 1.0]$, and $P_{allow} + P_{deny} = 1.0$. The uncertainty score is computed based on the *entropy* of the predicted class distribution: $Entropy = \sum_{i \in \{allow, deny\}} -P_i \log P_i$. A large entropy value indicates high uncertainty; entropy is minimized when P_{allow} or P_{deny} equals 1, which indicates that the probabilistic classifier is 100% sure about the class prediction.

[3]In principle, we can also use the *uncertainty score* to suggest to the user when it would be prudent to stop labeling.

After the sampling phase terminates, the *classifier construction phase* trains the classifier $\widehat{pref}$ using the labeled friends $F_{labeled}$.

Note that the classification algorithms used in the sampling phase and the classifier construction phase need not be the same [25]. We tried a variety of classifiers in our experiments. From a practical perspective, there may be additional considerations. If the sampling process is interactive, it is important that the classifier used in that phase be efficiently updatable; classifiers such as Naive Bayes appear to be a good option for that phase. In contrast, for typical-size friend lists, we do not expect performance to be much of a concern in the classifier-construction phase. For this part, user attention, rather than performance is the main bottleneck; in most cases, the classifier can be trained within a few seconds. As we will see in Section 4, if it is important to communicate the model $\widehat{pref}$ back to the user, then a human-readable classifier (e.g., decision tree) is attractive for the second phase.

3.4 Incremental Maintenance

Of course, users are always adding new friends. Suppose that the user has labeled an initial set of friends, using the active learning wizard described above. Ideally, we would like to satisfy the following two goals with respect to incremental maintenance:

1. When the user adds new friends, the classifier $\widehat{pref}$, which has been learned by the wizard, should make reasonable predictions for the new friends, without any additional input from user.
2. After the user adds new friends, the user may continue labeling friends. The wizard should use these new labels, in combination with the user's original input, without wasting the original labels.

Both of these goals are easily satisfied by the active learning wizard. Given the original set of friends F with a subset $F_{labeled}$ of them labeled, when some new set of friends F' is added, the privacy settings for the new friends can be predicted by constructing $\widehat{pref}$ using $F_{labeled}$, and applying it to each friend in F'.

The only part of this process that is tricky is managing features based on community structure. Recall that community-membership features are extracted from the labeled and unlabeled data. Thus, when new friends arrive, we will need to reconstruct the communities using $F \cup F'$. However, the labels that the user has assigned to individual friends remain valid. For example, in Figure 3, after new friends are added, the community structure may change (i.e., we may need to replace features $G_0, G_1, ...$). However, the label *allow* still applies to the (new feature-vector representation of) friend Alice Adams.

Finally, if new friends are added and the user wishes to devote more effort to refining her privacy settings, this is easy. The wizard simply adds F' to $F_{unlabeled}$, and continues the sampling process described in the last section.

4. MODEL VISUALIZATION AND MODIFICATION

The active learning wizard interacts with the user by asking her to label specific friends. This type of interaction is ideal for non-technical users, who have difficulty reasoning holistically about their policy configurations. On the other hand, the classifier and auto-configuration are essentially a black box, and more advanced users may want to understand the rationale behind the resulting configuration. For these users, we propose some additional tools that allow the user to visualize and update the classifier learned by the basic wizard.

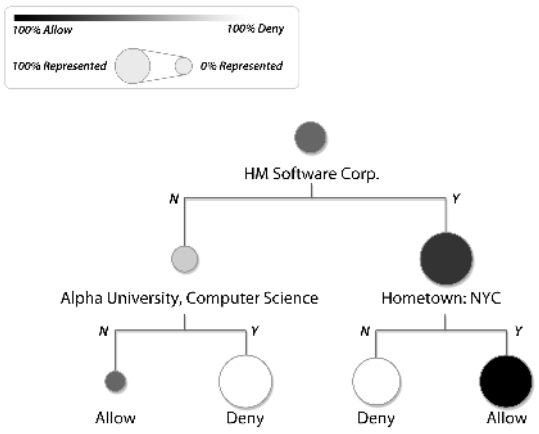

Figure 4: Visualization of Decision Tree Model

While the basic wizard can use any classification algorithm, if we are going to display the result to the user, then it is important to choose a classifier that is human-readable. Thus, in the remainder of this section, we will assume that, in the classifier construction phase, the active learning wizard constructs a binary decision tree.

4.1 Visualization

The basic structure of a binary decision tree is easily interpretable: Each interior node represents a binary condition (e.g., Hometown = NYC), and each leaf contains a decision (*allow* or *deny*). Each node (either interior or leaf) corresponds to a set of friends that are consistent with the binary conditions from root to the node. For the privacy-preference model, however, it is necessary to incorporate several additional pieces of information into the basic decision tree.

First, automatically-extracted communities (e.g., G_{20} in the running example) are meaningless to the user by default. Thus, in the visualization, we need to produce a meaningful description of each community. One reasonable option extracts unique keywords from the profiles of friends in each community (e.g., using the TF-IDF score).

Second, we observed that in some cases the resulting decision trees are large, and difficult to view all at once. To help guide users towards parts of the tree that are likely to require attention, we incorporate two additional pieces of information for each node:

- **Class Distribution:** For each node in the tree, the visualization indicates the class distribution (i.e., proportion labeled *allow* and *deny*) of the labeled friends who satisfy the conditions for the (subtree rooted at the) node.
- **Representative Rate:** the proportion of labeled friends among all friends who satisfy the conditions for a node.

EXAMPLE 4. *Figure 4 shows a decision tree that was trained using User* K*'s privacy preferences for Date of Birth. For each node, the class distribution is shown in grayscale, and the representative rate is indicated by node size. For example, the diagram indicates that friends who are members of*

HM Software Corp., and who have hometown NYC should be allowed to see Date of Birth. The representative rate is high, meaning that the user has explicitly labeled most of these friends. In contrast, notice that there is another node on the left, which describes all friends who are not part of HM Software Corp., and also not part of Alpha University Computer Science. Few friends in this category have been labeled, as indicated by the small circle. Further, the class distribution is heterogeneous.

4.2 Modification and Incremental Maintenance

The model visualization may guide the user in determining which friends need further attention. After visualizing the model, the user may decide to label more friends. She can do this by choosing a node (often one with low representative rate) and labeling more friends in the node. We do this as follows: When the user clicks on a node (interior or leaf), the unlabeled friends in the subtree rooted at that node are shown to the user in order of decreasing uncertainty (defined in Section 3.3). The user could label some of the displayed friends, and the visualization would change accordingly.

Incremental maintenance of model visualization and modification is straightforward and does not require additional effort. After new friends come, they are added to corresponding nodes in the tree. Class distribution will remain the same while the representative rate would decrease. The modification process remains the same: unlabeled friends (including the newly added friends) of a node would be displayed to the user when a node is clicked.

5. EVALUATION

The goal of our experiments is to analyze the effort-accuracy tradeoff achieved by our privacy wizard. Specifically, we want to answer the following two questions:

- How effective is the active learning wizard, compared to alternative policy-specification tools?
- Which features (e.g., community structure, profile information, etc.) are the most useful for predicting privacy preferences?

To answer these questions, we collected raw privacy preference data from a population of real Facebook users. Our results indicate that the active-learning wizard is more effective than existing alternatives. The results also indicate that automatically-extracted communities are very effective features for predicting privacy preferences.

5.1 Collecting Preference Data from Real Users

As the basis for our evaluation, we collected detailed privacy preference information from a group of real social network users. We conducted our study electronically using Facebook. We selected Facebook in particular because of the availability of an open development platform [1], and we built a Facebook application, which allowed our study subjects (a set of Facebook users) to exhaustively label their privacy preferences for all of their friends.

Our application presented each study subject with two questionnaires. The first questionnaire consisted of a series of coarse-grained questions, which asked, for each profile data item, whether the user would like to share the data item with *all friends*, *some friends*, or *no one*. For the purpose of the user study, we selected a representative set of

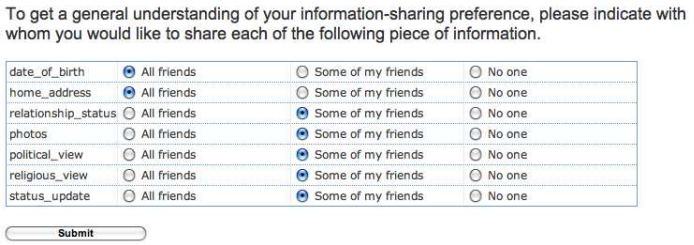

Figure 5: Screenshot of user study application, general questions

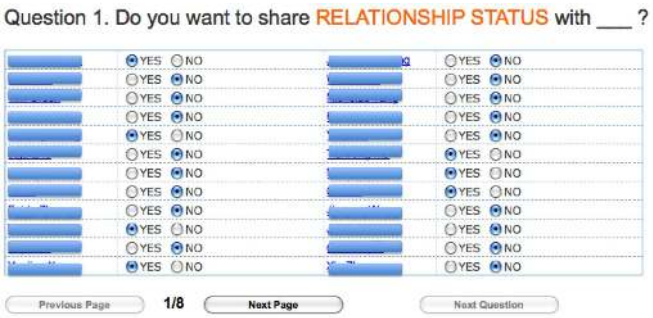

Figure 6: Screenshot of user study application, detailed questions.

profile data items: *Date of Birth*, *Home Address*, *Relationship Status*, *Photos*, *Political Views*, *Religious Views*, and *Status Updates*. A screenshot of the first questionnaire is shown in Figure 5.

The second questionnaire collected more detailed information. For each profile data item for which the user selected *some friends* during the first phase, we solicited detailed information during the second phase. The questionnaire listed the user's friends in a random order, and for each friend f, we asked the user to indicate her preferred access level for the friend: YES (interpreted as *allow*), NO (*deny*). The friends were presented in a sequence of pages, with 24 friends per page. A screenshot of the second questionnaire is shown in Figure 6. (The names of the user's friends have been hidden for confidentiality.)

In addition to the privacy preference information, the Facebook application allowed us to view the information about each subject's neighborhood described in Section 3.2.

A total of 45 people participated to our user study by labeling preferences. Of the 45 respondents, 27 of them were male, and 18 of them were female. The respondents are primarily the authors' colleagues, and they volunteered to participate. Our respondents had an average of 219 friends. The maximum number of friends was 826 and the minimum number of friends was 24. During the first phase, 30 of the respondents indicated that at least one profile data item should be visible to *some friends*. In total, there were 64 $(user, data\ item)$ pairs that required fine-grained privacy controls; that is, the users specified that these data items should be visible to *some friends*.

5.2 Experimental Setup

Our experimental setup incorporated several open-source packages. For community-finding, we used the implementation of the edge-betweenness in the *iGraph* library [3]. (We modified the algorithm as described in Section 3.2.1 to find hierarchical communities.) For classification, we used the

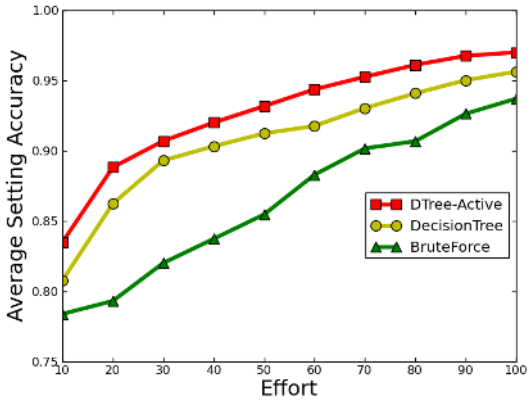

Figure 7: Effort vs. Average Accuracy tradeoff (within limited effort 100)

NaiveBayes, *NearestNeighbors*, and *DecisionTree* operators found in the *RapidMiner* [31] package.

5.3 Comparing Policy-Specification Tools

Our first set of experiments compares the active-learning wizard with alternative policy-specification tools. Because our other experiments (Section 5.4) show that community-based features are extremely effective, we use these features for the experiments in this section. We include results for the following three approaches:

- **DTree-Active:** This is a specific implementation of the active learning wizard described in Section 3. We used a Naive Bayes classifier in the sampling phase, and a decision tree to construct the final classifer.
- **DecisionTree:** To isolate the effects of the uncertainty sampling, we have also implemented a strawman solution. Whereas DTree-Active selects samples based on an uncertainty estimate, this algorithm selects samples at random. Like the previous approach, it uses the labeled examples to train a Decision Tree classifier.
- **BruteForce:** As a baseline, we evaluated a strawman policy-specification tool based on the following process: The user selects a default setting (in our experiments, this is assumed to be the majority class label). Then, the user can assign labels, one-by-one, to friends. Any friend left unlabeled is given the default label. The effort required by this process is very similar to the effort required to manually assign friends to lists, as required by the Facebook policy-specification tool.

In addition to the three tools described above, we also evaluated some variations of the active-learning and random-sampling wizards. In particular, we tried using alternative classifiers (Naive Bayes, K-Nearest Neighbors, and Decision Trees) for both sampling and classifier construction. The results were quite similar, and they are omitted for space.

5.3.1 Static Case

We begin with the static case, where the user is constructing a policy from scratch for a static set of friends. As the user applies more effort (i.e., labels more friends), using each of the policy-specification tools, we expect that the user's setting accuracy will increase.

Figure 7 illustrates this effort-accuracy tradeoff in a very rough way. The x-axis shows the number of friends labeled (up to 100), and the y-axis shows the average setting accuracy. (This is the average across all 64 *(user, data item)* pairs for which we obtained detailed preference information in our user study.) As expected, the active-learning approach (DTree-Active) outperforms the random-sampling approach (DecisionTree), and both outperform BruteForce. The results for DTree-Active are promising from a practical perspective, too; by labeling just 25 friends, users achieve an average setting accuracy of over 90%.

Of course, by averaging across different users and data items, Figure 7 does not capture all of the interesting details of the comparison. To understand the results better, we also developed a scoring approach. Intuitively, for a particular *(user, data item)* pair, the score S_{static} is a real number in $[0, 1.0]$ that measures the normalized area beneath the effort-accuracy curve; higher scores are better.

DEFINITION 1 (STATIC SCORE). *For a particular* user *and* data item*, the effectiveness of a policy-specification tool can be summarized using a score, where* $Accuracy_F(E = e)$ *is the setting accuracy achieved after applying effort* e *on the set of friends* F*:* $S_{static} = \frac{\sum_{e=0}^{|F|} Accuracy_F(E=e)}{|F|}$.

Using this scoring mechanism, our results are summarized in Figure 8, which shows the mean S_{static} score, as well as the standard deviation, across all 64 *(user, data item)* pairs:

Tool	S_{static}	
	mean	std
DTree-Active	0.94	0.04
DecisionTree	0.92	0.05
BruteForce	0.88	0.08

Figure 8: Comparison Summary (Static Case); Difference between tools is statistically significant based on a paired t-test

For each *(user, data item)* pair, we obtained a S_{static} score for each alternative policy-specification mechanism. Observe that, for example, the scores obtained for user Bob's Date of Birth using DTree-Active and DecisionTree can be treated as a dependent pair. Thus, we can test whether, for example, the S_{static} score for DTree-Active is significantly better than the score for DecisionTree using a paired-sample t-test. After performing this test, we discovered that, while the mean scores are similar, the differences between the policy specification tools are statistically significant. (DTree-Active is superior to DecisionTree, which is superior to BruteForce.)

Finally, while the results are omitted for space, we observed that for other classifiers the results are similar (i.e., active learning is superior to learning from a random sample, which is superior to the brute force approach).

5.3.2 Dynamic Case

The previous experiments focused on policy-specification for a static set of friends. In this section, we continue comparing the three policy-specification tools, but this time in the dynamic case, where the user adds new friends over time. In the following, we will denote the initial set of friends F, and suppose that the user adds a new set of friends F'.

To capture the dynamic case, we extend the scoring approach described in the previous section. Specifically, we will use two new scores: S_{pred} and $S_{dynamic}$.

The first score (S_{pred}) is based on the following scenario. Using one of the policy-specification tools, the user assigns

labels to e of the friends in the original set F (i.e., expends effort e). Then, the new set of friends F' arrives, and we measure the setting accuracy for the *new* friends,[4] which we denote $Accuracy_{F'}(E = e)$. Like before, for a particular *user* and *data item*, we will measure this across all values of e, and summarize the result with a single score.

DEFINITION 2 (PREDICTION SCORE). *The prediction quality of a policy-specification tool can be summarized using the following score. $Accuracy_{F'}(E = e)$ is the predictive accuracy of the settings, trained using effort e, and applied to a* new *set of friends F': $S_{pred} = \frac{\sum_{e=0}^{|F|} Accuracy_{F'}(E=e)}{|F|}$.*

The second score ($S_{dynamic}$) is based on a slightly different scenario. In this case, we assume that the user labels E friends (from the original set F). Then, a new set of friends F' is added, and the user labels E' more friends. We will use the notation $Accuracy_{F \cup F'}(E = e, E' = e')$ to denote the resulting setting accuracy for the full friend set $F \cup F'$. In this case, we will measure the accuracy across all values of E and E'; $S_{dynamic}$ is a real number in $[0, 1.0]$.

DEFINITION 3 (DYNAMIC SCORE). *The effectiveness of a policy-specification tool in a dynamic setting can be summarized using the following score:*

$$S_{dynamic} = \frac{\sum_{e=0}^{|F|} \sum_{e'=0}^{|F'|} Accuracy_{F \cup F'}(E=e, E'=e')}{|F||F'|}.$$

Using both of these scoring mechanisms, our results are summarized in Figure 9. In order to simulate the case of adding new friends, for each user, we randomly pick 30% of their friends as new friends while the remaining are regarded as the original friends. Again, based on a paired test, we also observe that DTree-Active is significantly better than DecisionTree, which is significantly better than BruteForce.

Tool	$S_{dynamic}$		S_{pred}	
	mean	std	mean	std
DTree-Active	0.92	0.05	0.82	0.15
DecisionTree	0.90	0.06	0.81	0.13
BruteForce	0.87	0.10	0.74	0.18

Figure 9: Comparison Summary (Dynamic Case); Difference between tools is statistically significant based on a paired t-test

5.3.3 Impact of Class Distribution

In our final set of comparison experiments, we observe that it is common for different users to have different distributions of privacy preferences. For example, user A may *allow* 90% of his friends to view Date of Birth, while user B may assign only 40% of friends *allow* permission. Ideally, we should adopt a policy-specification tool that adapts to these differences.

In order to measure the effect of skewed class distribution on each of the policy-specification tools, we use p to represent the proportion of labels in the minority class (i.e., $0 \leq p \leq 50\%$), and we partition our experimental data into three groups according to p value: $p \in (0\%, 10\%]$, $p \in (10\%, 30\%]$ and $p \in (30\%, 50\%]$.

[4]For DTree-Active and DecisionTree, the settings for the new friends are obtained by applying the classifer to F'. The best BruteForce can do is assign each of the new friends the default label.

Figure 10 summarizes the results, using the S_{static} score. In cases where p is low (i.e., users have homogeneous preferences for all friends), the improvement from using the active learning wizard is small. However, when the p value is larger (e.g., $p \in (30\%, 50\%]$), the active learning wizard is particularly helpful.

Tool	$p \in (0\%, 10\%]$		$p \in (10\%, 30\%]$		$p \in (30\%, 50\%]$	
	mean	std	mean	std	mean	std
DTree-Active	0.95	0.04	0.93	0.02	0.92	0.06
DecisionTree	0.95	0.03	0.90	0.03	0.88	0.04
BruteForce	0.94	0.04	0.88	0.07	0.80	0.05

Figure 10: Effects of class distribution (S_{static} score)

5.4 Comparing Features

Our final set of experiments compares the effectiveness of different alternative features, which can be used by learning-based wizards. In preliminary studies, we observed that DTree-Active, which uses a Naive Bayes classifier during the sampling phase, and then constructs a DecisionTree clasifier using the labeled data, resulted in the highest accuracy of all our active learning wizards (by a slight margin). Thus, in this section, we will present results based on the DTree-Active tool.

We compared five different combinations of features: (For more details, see Section 3.2.)

- *Community* These experiments used only features based on extracted communities.
- *Profile* These experiments used only profile-based features such as gender, age, education history (high school and college), work history, relationship status, political views, and religious views.
- *Activity* These experiments used only features based on online activities such as Facebook groups, "fan" pages, events, and tagged photos.
- *None-Comm* These experiments used only Profile and Activity features.
- *All* These experiments used all of the above.

Figure 11 summarizes our results. In addition, as described in the last section, we also conducted a paired sample t-test for each of the scores S_{static}, $S_{dynamic}$ and S_{pred}). We observed that *Community* is statistically significant better than all other feature combinations. It's interesting to notice that features as profiles and online activities are not helping much, it may be partially because that these features are usually incomplete and they're also conjecturable from the community features.

6. RELATED WORK

The development of usable, fine-grained tools for protecting personal data is a serious emerging problem in social media [4, 19, 21, 22, 23, 36]. In one study, Acquisti and

Features	S_{static}		$S_{dynamic}$		S_{pred}	
	mean	std	mean	std	mean	std
Community	0.94	0.04	0.92	0.05	0.82	0.15
Profile	0.87	0.07	0.84	0.08	0.67	0.15
Activity	0.89	0.07	0.88	0.08	0.78	0.16
None-Comm	0.87	0.06	0.85	0.07	0.70	0.15
All	0.92	0.05	0.89	0.06	0.78	0.13

Figure 11: Comparing features (DTree-Active)

Gross discovered that while users of social networking sites (Facebook, MySpace, Friendster, etc.) expressed high levels of concern about their privacy, the same users often did not implement strict privacy controls over their profiles. In many cases, this appeared to be due to users' poor understanding of the available privacy controls and the visibility of their profiles [4, 22].

Several recent papers have proposed novel user interfaces for specifying Facebook-style privacy settings, but none has constructed a wizard of the style described in this paper, which models and anticipates a user's preferences based on limited user input. Most related to our work is a pair of proposals by Adu-Oppong et al. [5] and Danezis [14]. Both propose partitioning a user's friends into lists, based on communities extracted automatically from the network, as a way to simplify the specification of privacy policies. ([14] describes this partitioning as a way of inferring a privacy "context.") While both are related to our work, neither studies real users' privacy preferences to evaluate their proposal. Also, in both cases, the proposed tools are based on partitioning friends into a fixed set of non-overlapping communities, which does not resolve the challenge of community granularity.

In a mobile location-based application, Ravichandran et al. [34] studied the problem of predicting a user's privacy preferences (i.e., share her location or not) based on location and time of day; however, this work did not consider taking an underlying social network structure into account when making these decisions.

After a policy is specified, many have observed that it is important to provide tools to help users understand the resulting settings. Lipford et al. proposed and evaluated an "audience view," which allows a user to view her profile as it appears to each of her friends [27]. A variation of this interface appears to have been recently adopted by Facebook. This work is quite complimentary to ours; while the audience view helps a user to understand and evaluate the correctness of an existing policy, it does not assist the user in creating the policy in the first place.

In a similar vein, recent work has proposed a methodology for quantifying the risk posed by a user's privacy settings [30, 28]; at a high level, a risk *score* communicates to a user the extent to which his privacy settings differ from those of other users who are close to him in the social network graph. Like the audience view, the score provides feedback to the user regarding his existing settings, but it does not help him in creating an initial policy. Further, the tools only provide a single score, so if a user's privacy settings are out of line, they do not communicate to the user precisely how he should refine his settings in order to achieve a more acceptable configuration.

Fong et al. [17] and Carminati et al. [11, 12] look to formalize the access control model required by social networking sites. Our work is complementary; the goal is to assist users in expressing their privacy preferences.

In this paper, we have focused on helping users to express simple privacy settings, which is a difficult task on its own. We have not considered additional problems such as inference [40], or shared data ownership [38]. As a simple example of the former, suppose that a user Alice wishes to hide her political affiliation. The first problem, which is the focus of this paper, is to make sure that Alice can even express this preference to the social networking site. However, even if the site hides Alice's political affiliation, it may still be possible for an attacker to infer the hidden information [40]. (For example, if 95% of Alice's friends are liberal, then there is a good chance that Alice is also liberal.) Interestingly, it is often not possible for Alice to prevent this kind of inference by simply configuring her own privacy settings. The PrivAware system [8] makes an initial step toward quantifying the risk of this type of inference; as a solution, the authors suggest removing certain friend relationships to reduce the inference risk.

Broadly-speaking, social networking websites have led to a number of interesting research questions in information security and privacy. For example, in 2007, Facebook opened a development API, which allows developers to construct their own applications leveraging user profile data [1]. This was met with some concern for personal privacy; for example, one study revealed that applications written using this API could often access significantly more information than necessary for their core functionality [16]. As an initial solution to this problem, Felt and Evans proposed a proxy-based architecture, which limits the amount of information available to installed applications [16]. Singh et al. propose a trusted third-party mediator called xBook [37]. Lucas and Borisov [29] and Anderson et al. [6] consider an even more restrictive case in which users are reluctant to share their personal information with the Facebook service.

Social networking sites may also enable new forms of classical attacks, including phishing [9] and spam [10]. [15] considers the new risk to anonymous routing that is posed by an attacker who knows users' social network graphs.

Finally, recent work has focused on the privacy risks associated with publishing de-identified social network graphs for research. Even if all profile information is removed, it is often possible to re-identify individuals in the published data simply based on unique graph topologies [7, 24, 32].

7. CONCLUSION AND FUTURE WORK

Privacy is an important emerging problem in online social networks. While these sites are growing rapidly in popularity, existing policy-configuration tools are difficult for average users to understand and use.

This paper presented a template for the design of a privacy wizard, which removes much of the burden from individual users. At a high level, the wizard solicits a limited amount of input from the user. Using this input, and other information already visible to the user, the wizard infers a privacy-preference model describing the user's personal privacy preferences. This model, then, is used to automatically configure the user's detailed privacy settings.

To illustrate this idea in concrete terms, we have built a sample wizard, which is based on an active learning paradigm. We have also constructed a visualization tool, which allows advanced users to view and modify the resulting model. Our experimental evaluation, which is based on detailed privacy-preference information collected from 45 Facebook users, indicates that the wizard is quite effective in reducing the amount of user effort, while still producing high-accuracy settings. The results also indicate that the community structure of a user's social network is a valuable resource when modeling the user's privacy preferences.

In the future, we plan to conduct more user studies to understand how users like the wizard comparing to alternative privacy settings tools, and how much time users are

willing to put into the policy specification process. We will also consider other instances of privacy wizards. For example, our active learning wizard solicits user input in a very simple form (i.e., asking the user to assign a label to a *(friend, data item)* pair), which is easy for the user to understand. Perhaps there are other questions that would yield more information, or require less user effort. Also, in this work, we considered three main sources of information in a user's neighborhood when constructing the privacy-preference model: communities, profile data and online activities. In the future, other sources of information may be taken into account. For example, it would be interesting to understand whether ideas such as *tie strength* [20] are useful in predicting privacy preferences.

Acknowledgements

This work supported in part by NSF grants IIS-0438909 and CNS-0915782. The authors would like to thank Lada Adamic for her help in understanding the complex networks literature, and H.V. Jagadish and Alex Halderman for their comments on earlier versions of the paper.

8. REFERENCES

[1] Facebook development platform. http://developers.facebook.com/.

[2] Facebook statistics. http://www.facebook.com/press/info.php?statistics.

[3] The igraph software package for complex network research. *InterJournal Complex Systems*, 2006.

[4] A. Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Privacy Enhancing Technologies Workshop*, 2006.

[5] F. Adu-Oppong, C. Gardiner, A. Kapadia, and P. Tsang. Socialcircles: Tacking privacy in social networks. In *Symposium on Usable Privacy and Security (SOUPS)*, 2008.

[6] J. Anderson, C. Diaz, J. Bonneau, and F. Stajano. Privacy-enabling social networking over untrusted networks. In *WOSN*, 2009.

[7] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.

[8] J. Becker and H. Chen. Measuring privacy risk in online social networks. In *Web 2.0 Security and Privacy Workshop*, 2009.

[9] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: Automated identity theft attacks on social networks. In *WWW*, 2009.

[10] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social networks and context-aware spam. In *CSCW*, 2008.

[11] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *Workshop on Reliability in Decentralized Distributed Systems*, 2006.

[12] B. Carminati, E. Ferrari, and A. Perego. Private relationships in social networks. In *ICDE Workshops*, 2007.

[13] L. Church, J. Anderson, J. Bonneau, and F. Stajano. Privacy stories: Confidence on privacy behaviors through end user programming. In *Symposium on Usable Privacy and Security (SOUPS)*, 2009.

[14] G. Danezis. Inferring privacy policies for social networking services. In *AISec*, 2009.

[15] C. Diaz, C. Troncoso, and A. Serjantov. On the impact of social network profiling on anonymity. In *Privacy-Enhancing Technologies Workshop*, 2008.

[16] A. Felt and D. Evans. Privacy protection for social networking platforms. In *Web 2.0 Security and Privacy Workshop*, 2008.

[17] P. Fong, M. Anwar, and Z. Zhao. A privacy preservation model for facebook-style social network systems. University of Calgary Technical Report 2009-926-05, 2009.

[18] S. Fortunato. Community detection in graphs. http://arxiv.org/abs/0906.0612v1 (Preprint), 2009.

[19] C. Gates. Access control requirements for web 2.0 security and privacy. In *Web 2.0 Security and Privacy Workshop*, 2007.

[20] E. Gilbert and K. Karahalios. Predicting tie strength with social media. In *CHI*, 2009.

[21] K. Gollu, S. Saroiu, and A. Wolman. A social networking-based access control scheme for personal content. In *SOSP*, 2007.

[22] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Workshop on Privacy in the Electronic Society*, 2005.

[23] M. Hart, R. Johnson, and A. Stent. More content - less control: Access control in the web 2.0. In *Web 2.0 Security and Privacy Workshop*, 2007.

[24] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. In *VLDB*, 2008.

[25] D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *ICML*, 1994.

[26] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *SIGIR*, 1994.

[27] 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.

[28] K. Liu and E. Terzi. A framework for computing the privacy scores of users in online social networks. In *ICDM*, 2009.

[29] M. Lucas and N. Borisov. flybynight: Mitigating the privacy risks of social networking. In *Workshop on Privacy in the Electronic Society*, 2008.

[30] E. M. Maximilien, T. Grandison, T. Sun, D. Richardson, S. Guo, and K. Liu. Privacy-as-a-service: Models, algorithms, and results on the facebook platform. In *Web 2.0 Security and Privacy Workshop*, 2009.

[31] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *SIGKDD*, 2006.

[32] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, 2009.

[33] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, 69(2), 2004.

[34] R. Ravichandran, M. Benisch, P. Kelley, and N. Sadeh. Capturing social networking privacy preferences. In *Symposium on Usable Privacy and Security (SOUPS)*, 2009.

[35] 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.

[36] D. Rosenblum. What anyone can know: The privacy risks of social networking sites. *IEEE Security and Privacy*, 2007.

[37] K. Singh, S. Bhola, and W. Lee. xBook: Redesigning privacy control in social networking platforms. In *USENIX Security*, 2009.

[38] A. C. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *WWW*, 2009.

[39] 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.

[40] E. Zheleva and L. Getoor. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *WWW*, 2009.