Learning Privacy Rules Cooperatively in Online Social Networks*

Berkant Kepez
Department of Computer Engineering
Bogazici University
Istanbul,Turkey
berkant.kepez@boun.edu.tr

Pınar Yolum
Department of Computer Engineering
Bogazici University
Istanbul,Turkey
pinar.yolum@boun.edu.tr

ABSTRACT

The use of online social networks is growing rapidly. With this rapid increase, preserving privacy of users is becoming harder and harder. Typically, social networks address the privacy problem by asking users to define their privacy constraints up front. However, many times deciding on whom to show a post is dependent on the post itself and its context. Hence, users are forced to configure each post specifically, which is both cumbersome and prone to error. Accordingly, this paper first proposes an approach that suggests privacy configurations for each post. The suggestions are based on learning from users' previous posts and configurations. However, when the user does not have many previous posts, recommendations need to take other information into account. We propose a multiagent system architecture where agents of the users consult other users' agents about possible privacy rules they can take into account.

CCS Concepts

•**Security and privacy** → **Social network security and privacy;**
•**Computing methodologies** → **Multi-agent systems;**

1. INTRODUCTION

Privacy has long been accepted as an important concept in developing and running software. A typical software publishes its privacy agreement, which a user accepts. A similar pattern applies to online social networks, with additional settings for customizing the policy. For example, a user can choose to share content only with her friends, whereas another user may choose to share content with everyone in the system. While customization enables users to configure their privacy better, it also brings a burden on the user as she needs to decide for each post who should be in the audience [8]. This creates additional load on the users as well as introduces a factor of human error. For a person that shares content frequently, it is easy to forget to add or remove a certain individual from a post's audience.

*This work has been supported by TUBITAK.

PrAISe '16, August 29-30, 2016, The Hague, Netherlands

DOI: http://dx.doi.org/10.1145/2970030.2970036

EXAMPLE 1. *Alice travels to New York. When she visits touristic places, she likes to share them on her online social network account. She plans to visit her friends, Bob and Carol, who live there, too. However, she wants the visit to be a surprise. Thus, she wants her friends not to see her posts. She successfully hides various posts from Bob and Carol by removing them from the post's audience. However, she forgets to set it correctly when she visits Statue of Liberty. Both Bob and Carol see the post and the surprise is ruined.*

Ideally, if Alice's software could learn over time that Alice is not sharing her recent posts with Bob and Carol and if it could warn Alice accordingly, it would be of tremendous help. We realize this by employing machine learning (ML) techniques. The idea is that Alice's posts are classified over time based on whether they are being shown to certain individuals (in this case Bob and Carol). When Alice is in a position to share a post with others, the classifier first checks whether this would cause any harm based on previous sharing behavior. If so, the system interrupts upload and asks for permission.

When a user has enough posts in the system, learning from previous posts is a good direction. However, many users face privacy problems more when they first start to use a system, since the implications of their action on the system are not clear. Consider the following example:

EXAMPLE 2. *Charlie is a new user of the online social network. He does not know what types of things he needs to consider to preserve his privacy. When he has a post, he would like to get others' opinion about how he should set the audience of the post.*

In such cases, we need to have a software that can help users in Example 2. One way of approaching this problem is from a system's point of view. That is, if overall the system data are available, one can build clustering systems so that a user can be recommended privacy policies based on what other users are doing. However, when such data are not available, then the user is left to her own resources to figure out. In offline world, users ask each other for help. Here, we mimic a similar approach in a multiagent setting, where a user in need of privacy recommendations turns to others in the system for help. The user's agent queries other agents, collects their recommendations, and consolidates them for a final result. We study various ways to realize this.

2. SUGGESTING PRIVACY CONFIGURATIONS: AN ML APPROACH

We envision a system where each user is represented by a software agent. An agent is a computation that can perceive, reason,

act and communicate with other agents [6]. The agent's broad aim is to help its user manage her privacy. Contrary to current social network applications, where humans are the only users of the system, we envision a system where the user actions are supported by the agent. For example, when a user is about to upload a photo, her agent can warn her about particular consequences (e.g., an unintended audience will see the post because the audience is set to public). The user is free to override the agent.

The agent has three main sources of information, on which it can act. The first source is the user's own privacy rules. If the user already has predefined privacy rules, then enforcing them over a post is easy. Since this case is generally straightforward, we do not explicitly address this.

The second case is where the user might not have specified its privacy rules explicitly but has shared many posts in the system. A user that enters the system can share posts as she sees fit and she can set the audience of the posts.

Since we are concerned with the individuals to whom the post is not intended to be shown, the agent can track cases where a user has explicitly removed a certain individual from an audience and learn over time a privacy policy for the user.

Following Example 1, this would correspond to Alice sharing a post of Brooklyn Bridge and restricting Bob and Carol. While other individuals are allowed to see the post, Alice's agent records the fact that Alice has disallowed Bob and Carol. The idea then is to generalize whom the posts are not shown to.

Machine Learning techniques are well-suited for the task of learning privacy preferences of the user. The idea is to learn whom a given type of post is shared with based on the audience of the post. The techniques which are used in our study are Decision Trees, Random Forests (RF), Support Vector Machines (SVM), Naive Bayes (NB), and Extreme Learning Machines (ELM). Decision trees are trees of which each interior node represents a feature of an instance, each of their branches represents a value range for that feature, and each leaf node represents the class labels [3]. To generate the tree, ID3 algorithm has been used [10]. Random Forest (RF) consists of a collection of decision trees generated with random subsets of the original training data or random subsets of the input features [2]. Then, the final label is determined by majority voting of those decision trees. An SVM model is built by finding an hyperplane which minimizes the functional margin of the training data [12]. Then, this model is used to classify the test data. Naive Bayes classifier is one which assigns a class label to an instance based on its maximum likelihood using Bayesian-rule and independence assumption [7]. ELM is a feed forwarded neural network of which weights of hidden nodes are randomly assigned and output neurons are optimized with linear regression [5].

Each recorded post can be interpreted as a training data instance that is fed to a machine learning classifier. When converting the posts to the instances for the classifier, following attributes of posts are selected: the type, the sharing time, the location, the location context of posts and the ids of tagged friends in them. For each post, an instance with those features are created. Additionally, using the access rules, for each friend of the user, each instance is reproduced and their ids are added as a feature. Whether a friend is denied to access or not to a post is added into the instance as class label.

When user shares a post, our system intervenes the sharing process and suggests people who should be denied to access to that post. Then user decides the final access rule.

The third case is when the user has not shared many posts in the system. In that case, making a decision only based on the user's past posts will not yield accurate results. In that case, the agent can make use of the other agent's knowledge in the system to reach a decision. One approach to realize this could be if the other agents share their privacy rules with the agent.

However, sharing of privacy policies with others is another type of privacy violation for the other agents.

Another approach would be to ask others to classify a given post, which would not lead to privacy violation since the reasons would still be hidden. Thus, here we opt for such an approach. The agent then retrieves the results and makes a decision, as explained in Section 3.2.

In the first step, an instance of PostRequest is generated in the ontology with a unique identifier based on the time of generation. Then, elements of the post are created as their corresponding classes such as Location if they do not already exist in the ontology. Then they are linked with PostRequest or each other as properties by keeping the original structure of the post.

Size of Training (& Test dataset)	Accuracy
400 (400)	100.00
200 (200)	100.00
100 (100)	100.00
50 (50)	100.00
25 (25)	86.00
13 (13)	80.77
7 (7)	64.28

Table 1: The decrease in accuracy as the training data size decreases

3. EVALUATION

In the evaluation, a set of various agents in terms of privacy rules and the amount of decision noise are employed. These agents decide whether an access of an audience to a post should be accepted or not according to those privacy rules and then they flip their decision based on the probability determined by the amount of decision noise.

3.1 Single-Agent Results

In the first setting, we study a case where a single agent (e.g., Alice) uses her previous posts to predict the privacy status of a new post she is about to share.

In order to decrease the effect of structure of the training data on the evaluation results, 10 experiments were conducted for each machine learning algorithm. Each experiment was conducted with a new set of data for both training and testing. In each of them, the model is trained with 400 instances while it is tested again with 400 instances. Then, the accuracy of each ML method (such as SVM) is calculated as the average of accuracy of experiments of that method. A benchmark dataset was created based on content from Flickr and Reuters.

Table 1 shows that when data set has at least 50 training items, the change of size of training data set does not affect (average) accuracy. However, below 50, it starts to decrease. If we examine those with lower accuracy than 100% further, it is observed that some of the individual experiments with lower accuracy are caused by the training data in only one class. In 2, 4, and 6 individual experiments of the experiments with, respectively, 25, 13, and 7 training instances, that is the situation. In that case, the SVM models trained predicts the instances as the only class which they have learned. This affects the accuracy. Since those cases are also possible in practice, their results have taken place here. In order to ignore their effect, the accuracy of the rest of the individual experiments are calculated, which can be seen in Table 2:

Size of Training (& Test dataset)	Accuracy
25 (25)	100.00
13 (13)	100.00
7 (7)	78.57

Table 2: The decrease in accuracy as the training data size decreases

Only the one with 7 training instances has the accuracy less than 100 percentage.

It has been widely known that users may end up sharing information that they think is private, possibly because they do no think about the consequences up front [4]. It is also common to regret such posts later[13]. Since a user does not always behave in parallel with her privacy preferences in her mind, we have added noise to our dataset in order to reflect that gap. The noise is added at the step of determination of decision of whether access for an OSN post by another user should be allowed or denied. Without noise, an agent evaluates a post by taking privacy rules of its user in the Semantic Web Rule Language (SWRL) format into account and accepts the result of reasoning of the post based on SWRL rules as the final decision. However, after adding noise, it decides in the opposite direction of the result of reasoning with a probability p_n. For example, consider adding the noise with the probability (p_n) 0.1 to Example 1. Then, Alice's agent tries to decide whether a post which has the location information USA should be shared with Bob or not. As a consequence of reasoning on Alice's rules, specifically on the rule of not sharing USA posts with Bob and Carol, it concludes that the access for that post should be denied. However, with the probability 0.1, it allows the access.

To observe the effect of noise on the accuracy of classifiers, we have carried out again 10 experiments with the dataset with 400 training and 400 test instances. Accuracies are again calculated as the average of 10 experiments for each pair of classifier and noise value. As noise values, 0.05, 0.1, 0.2 and 0.4 has been employed. Furthermore, the case of 0 noise (no noise) has been also added in order to develop a comparison of the experiments with the noise, with the previous experiments, which are the ones without noise. Figure 1 shows that the accuracy of each classifier decreases as the noise increases. This is expected. However, up to p_n=0.1, except those of SVM and slightly ELM, none of the accuracies of classifiers are acceptable. They are below 85%. After p_n=0.1, none of the classifiers perform well. At that point, there is a trade-off between noise value and accuracy. For the sake of high accuracy, the gap between behaviors and expectations of a user, thus, noise cannot be ignored and cannot be assumed as 0. However, after a point, it may not reveal the actual influence of the gap. When p_n=0.1, the accuracy of SVM does not decrease below the 90%. Thus, we employ a noise value of 0.1 in the following experiments.

3.2 Multi-Agent Results

As can be seen in Table 2, when the number of training data instances decreases, the accuracy of predicting the label also decreases. When an agent has been trained with a few instances, the agent can guess that its predictions will not be too accurate. In such cases, the agent can use (some of) the other agents to reach a decision.

In this evaluation, we have created six more agents that make up the multiagent system. Bob has exactly the same privacy requirements as Alice. Carol's privacy requirement is more restrictive. She denies posts from both USA and Canada. Dave denies posts from Canada and Japan. The fourth agent is a random agent that picks randomly what the outcome will be. The last two agents are YES and NO agents. The former always allows posts and the latter always denies them. Note that Alice is not aware of these behaviors. For those multi-agent experiments, datasets for each agent were created in the same procedure with the single-agent experiments. The noise probability, which is used when determining the decision of sharing, has been chosen as 0.1 due to the reason mentioned in the Section 3.1. SVM has been selected as the classification method since it had the highest accuracy in the single-agent experiments. Alice's agent has been trained with 14 instances while agents of Bob, Carol and Dave have been trained with 100 instances. In the first experiment, Alice asks all the agents for the first of her test instances and gets the results. Since the answers vary and Alice has no other means to decide whom to trust, she applies majority voting where every agent's vote is equally weighted.

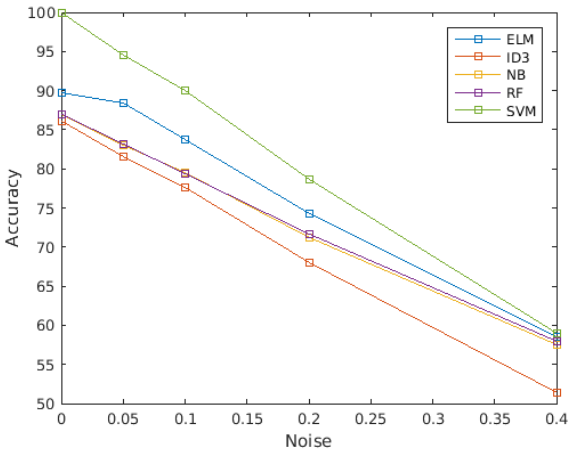

Figure 1: The effect of noise on the accuracy of each classifier

$$trusts_{a,i} = \begin{cases} 0 & \text{if } i \text{ equals } 0 \\ trusts_{a,i-1} + 1 & \text{if } a \text{ labels } i\text{th instance} \\ & \text{as labeled by the owner} \\ trusts_{a,i-1} & \text{otherwise} \end{cases} \quad (1)$$

Then, the trust levels are calculated with the Equation 1, where *i* is the number of support posts and *a* represents the agent to whom Alice asks. In that way, the original labels by Alice and the labels by other agents are compared, which are handled by second and third conditions in the equation. These trust values, $trusts_{a,i}$, are returned directly by the method CALCULATETRUST and assigned to $trusts$ matrix (line 2). Then, *responses*, the list which stores the responses of those agents for the test instance, and *results*, the list which stores the final collective results for the instance, are initialized via INITLIST, which returns a list filled with zeros in the given length (line 3-4). Since she has trust values for each agent, she firstly gets the responses of other agents for *ins* and stores them in *responses* (lines 5-7). For each case with different number of support posts (lines 8-20), she calculates the total weight, *totalweight*, by summing the trust values for each agent, and the total response, *totalresponse*, by summing the weighted responses of each agent for that instance, which are weighted by agents' trust values (lines 9-14). Following it, Alice's agent calculates a weighted average response by dividing them and it is compared with 0.5. If it is higher than 0.5, then result in that case is assigned as 1 (YES), otherwise 0 (NO) (lines 15-19). At the end, *results* are returned.

Algorithm 1: COLLECTIVEDECISION($owner$,n,ins)

Input: $owner$, the post owner
Input: n, the max number of support posts of $owner$
Input: ins, the instance to be classified
Output: $results$, the labels of ins for the cases with 1 to n support posts

$A \leftarrow owner$.getConnections();
$trusts \leftarrow$ calculateTrust($owner$,A);
$responses \leftarrow$ initList($A.length$) ;
$results \leftarrow$ initList(n) ;
foreach *a in A* **do**
  $responses[a] \leftarrow a$.classify($ins$);
end
for $i \leftarrow 1$ **to** n **do**
  $totalweight \leftarrow 0$;
  $totalresponse \leftarrow 0$;
  foreach *a in A* **do**
    $totalresponse$ += $trusts[a][i] * responses[a]$;
    $totalweight$ += $trusts[a][i]$;
  end
  if *totalresponse / totalweight > 0.5* **then**
    $results[i] \leftarrow 1$;
  else
    $results[i] \leftarrow 0$;
  end
end
return *results*

Our preliminary results indicate that agent s need a few interactions to identify the trustworthy agents and then successfully choose agents for the remaining interactions.

4. DISCUSSION

We have developed a framework in which agents can learn their users' privacy constraints. While it focuses on learning the privacy constraints regarding posts (including web links, location information, texts, multimedia content such as photos and videos), relevant works in the literature focus on those regarding only location, or only image.

Squicciarini *et al.* propose a privacy policy inference mechanism of images which are shared by users on online social networks [11]. The mechanism classifies images based on their content and metadata. Then policy for an image is chosen by using the policies of other images in its class. Their approach generally assumes that a centralized system can be used to find related policies. In our case, each agent has access to its own data and can request a content to be classified from a selected set of other agents. This enables agents to track who has access to their policies.

Bilogrevic *et al.* develop an information sharing system regarding privacy of location of the user, people around her and her availability [1].

While they just take the privacy of location information into account, we consider the whole structure of a post as our privacy problem. Furthermore, we aim to help users with few data items by employing trust in a multi-agent setting.

In order to learn users' privacy preferences regarding location information and help users with no or few data, Mugan *et al.* also employ machine learning techniques [9]. Observing the patterns in the data gathered from participants of the study, the exact values of features of location shares are mapped to the categorical values determined by them, each combination of which creates a state. By applying decision trees on users' data, privacy policies of users, their sharing decisions for each state, is learned. By clustering policies and again applying decision trees on each cluster, they learn *default persona*s, which represents the common characteristics of users in terms of privacy. Compared to our approach, again they only focus on the location attribute while we focus on sharing posts which can have location information. Furthermore, they decreases the input space to a small value, while ours' input space depends on the training da taset.

Our work here opens up interesting directions for further research. One direction is to add inference to the work [8]. More specifically, in Example 1, Alice tries to achieve location anonymity by hiding her location explicitly from her friends. However, in reality, many other details could give away the fact that Alice is actually in the US. It would be ideal if the agent could make the necessary inferences to help the user manage her privacy.

5. REFERENCES

[1] I. Bilogrevic, K. Huguenin, B. Agir, M. Jadliwala, M. Gazaki, and J.-P. Hubaux. A machine-learning based approach to privacy-aware information-sharing in mobile social networks. *Pervasive and Mobile Computing*, 2015.

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees belmont. *CA: Wadsworth International Group*, 1984.

[4] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pages 71–80, 2005.

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[6] M. Huhns and M. P. Singh, editors. *Reading In Agents*. Morgan Kaufmann, San Fransisco, 1998.

[7] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[8] N. Kokciyan and P. Yolum. PRIGUARD : A semantic approach to detect privacy violations in online social networks. *IEEE Transactions on Knowledge and Data Engineering*, 2016. In press.

[9] J. Mugan, T. Sharma, and N. Sadeh. Understandable learning of privacy preferences through default personas and suggestions. Technical Report CMU-ISR-11-112, Carnegie Mellon University, School of Computer Science, 2011.

[10] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[11] A. C. Squicciarini, D. Lin, S. Sundareswaran, and J. Wede. Privacy policy inference of user-uploaded images on content sharing sites. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):193–206, 2015.

[12] V. N. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[13] Y. Wang, G. Norcie, S. Komanduri, A. Acquisti, P. G. Leon, and L. F. Cranor. I regretted the minute i pressed share: A qualitative study of regrets on facebook. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 10. ACM, 2011.