Social Spam Discovery using Bayesian Network Classifiers based on Feature Extractions

Dae-Ha Park
Division of Information Technology
The Cyber University of Korea
Seoul, South Korea
summer69@cyberkorea.ac.kr

Eun-Ae Cho
Visual Display Division
Samsung Electronics Co., Ltd.
Suwon, South Korea
ea.cho@samsung.com

Byung-Won On
Advanced Institutes of Convergence Technology
Seoul National University
Suwon, South Korea
bwon@snu.ac.kr

***Abstract* — People always communicate with each other through social networking services (SNSs). However they often receive various kinds of unwelcomed messages that can be requests from uncomfortable friends or may be advertisements. In this paper, we defined these messages as "social spams", and suggested new classification method to detect them. By characterizing the problem of discovering social spams which frequently occurs in current popular SNSs, we extracted and exploited novel features that had not shown in the existing E-mail or web spamming prevention techniques. Our proposal for collecting various features such as behavior, celebrity, trust, common interest, etc. could incrementally been updated for SNS users. We modified the existing well-known classification techniques such as Bayesian network classifiers (BNCs) to customize for SNS features. To make decision efficiently, we computed Katz or trust scores with only part of updated network topologies.**

***Keywords* — Social network service; privacy; social spam discovery; Bayesian network classifier; Katz score**

I. INTRODUCTION

According to a report [1], over the past 12 months more than four-fifths of social networking service (SNS) users reported that they had received unwanted friend requests, messages, or posts on their social or professional network profiles. While friend requests on their own seem innocuous enough, they are often just the first step towards whatever the spammer's intentions of malicious activities are, because they redirect users to phishing or malware sites or even just unsolicited advertisements [2]. Furthermore, users have received an average of 64 unwanted friend invitations, messages or postings in the last 12 months. And Two-thirds of users said they would consider switching social network services if spam became too frequent [1]. Therefore, it is very important issue to detect unwelcomed messages and control them adequately.

To solve this issue, so-called "social spams", we define a spam as any sort of unwanted message between two users in a social networking service.

Traditional works on email and web spamming problems are [3] [4] [5] [6] [7]. However, as a solution for the social spam detection problem, using well-known spamming techniques employed in different domains (e.g., E-mails or web) is not the best strategy. More specifically, the existing researches are not effective in this problem due to the following three characteristics of social spams:

- Since most messages make it hard to use content-based spam filtering schemes, which have effectively been employed in the E-mail spam problem, content-based filtering schemes are not sufficient to detect social spams. For example, a friend request often does not have any messages or contents. Even though it contains a few contents, it is mostly very general. They do not generally reveal spamming words like "Viagra" or "stock".
- Determining whether a friend request is a spam or not is difficult due to its subjectiveness and the nature of social networks. Allowing friend requests only from someone who are directly or indirectly associated might hurt an important goal of social networks, making new friends. For instance, a user in Korea may want to make friends with another random user in Japan who might like the friend request as she always has fancied Korea. On the other hand, a male who is a perfectly normal user in a social network might be seeking to make friends with good-looking females and be sending a bunch of unwanted friend requests or messages to random females. Those messages could or could not be spam to the receivers. Thus the spamness is a subjective measure in social networks.
- The traditional well-known classification techniques such as Naïve Bayes or Support Vector Machines (SVM) [8] are not effective. For instance, unlike the existing spamming problem (e.g., E-mail spam problem), there exist explicit correlations among features in our social spam problem. For example, a recipient may accept a friend request if she is aware that a sender is her friend's friend. Another recipient may accept a friend request by his personality (e.g., social or introvert). Some recipients may accept a friend request by both his personality and friendship. Due to such feature dependencies in our problem, Naïve Bayes would not work effectively. Since SVM is required to run the expensive training step, it would not be efficient in our problem.

To address these challenges, we exploit various features based on previous history of users' spams, user behaviors, common interests and groups, and network topology by

friendship or influence as well as content-based spam analysis. Then, we develop a well-designed classification model which combines the features collected and determines whether a friend-request is a spam or not. In particular, we focus on the Bayesian Network Classifier (BNC) which would be effective with features in which there are close correlations among features [9].

II. Feature Extractions

In our problem, we define a social network G with n nodes and edges, assuming that G is a large graph, consisting of hundreds of thousands of nodes. An edge stands for the friendship link between two nodes. In other words, when two users would like to make friends to each other, one user sending a friend request r denotes a sender x, while the other user receiving r from x denotes a recipient y. If y accepts r, the friendship link l is created between x and y. Each node has its profile $prof(x)$ consisting of word terms Γ_x describing personal information and interests (e.g., hobby and favorite sports) of user x. In addition, a spam is defined as any sort of unwanted friend request between x and y in a social networking service S.

For simplicity, let us assume that we extract the features f from the static training set $Tr[t-c,t]$ (e.g., the past c months from the current time t), and then when x sends r to y at t, our spam filtering system (as centralized) detects if x becomes a spammer *on the fly*.

- *Request Reject Ratio* (*RR*): $RR = \frac{Number\_of\_rejects}{Number\_of\_friend\_requests\_sent}$. This feature indicates how many friend requests sent by x were rejected by other nodes in the previous time $Tr[t-c,t]$. The x has the particular value of *Request Reject Ratio*.
- *Request Acceptance Ratio* (*AR*): $AR = \frac{Number\_of\_acceptances}{Number\_of\_friend\_requests\_received}$. This is the opposite feature to *Request Reject Ratio*. This is, how many friend requests received by y were accepted in $Tr[t-c,t]$. The y has the particular value of *Request Acceptance Ratio*.
- *Celebrity* (*CB*): How influent x is in G? This is based on the hypothesis that y is more likely to accept a friend-request from famous x rather than others. This can be measured by the Katz method [10] that computes the degree of influence of an actor in a social network. x has his Katz score, quantifying the importance of x in G.
- *Trust* (*TR*): How trustable x is in G? Starting from a set of non-spammers (as seed nodes), and propagating their trust scores through *G*, estimate the trust ranking score for x.
- *Personality* (*PS*): If the personality of y is social, she is more likely to accept the friend-request from x. If the number of neighbors $\geq \theta$, we define her personality as social or *extrovert* (1), while her personality is shy or *introvert* (0), otherwise. y has a binary value of 0 or 1.
- *Same Community* (*SC*): If x and y are in the same group C, y is more likely to accept the friend-request of x. Thus investigation is needed if a set of communities created by a certain community detection technique has the input of network topology $(x,y) \in C$. x and y have a binary value: the same community (x,y) (1) and different communities (x,y) (0).
- *Friend's Friend* (*FF*): Is x friend's friend of y? If x is the friend's friend of y, y is more likely to guarantee x. x and y have a binary value – Friend's friend (x,y) (1) and (0), otherwise.
- *Commonness* (*CM*): $CM = \frac{|\Gamma_x \cap \Gamma_y|}{|\Gamma_x \cup \Gamma_y|}$, where Γ_x is a set of interest-tags in $prof(x)$ and Γ_y is a set of interest-tags in $prof(y)$. This feature works based on the assumption that users in S usually make new friends to peers who are interested in common interests – hobby, occupation, and so on. x and y have the same similarity score of common interests between x and y.
- *Content Analysis* (*CA*): $CA = \frac{\{\text{term } \tau | \tau \in \text{Blacklist } B, \tau \in prof(x)\}}{|B|} = \sigma$ or $\frac{\sigma}{\max \sigma \text{ in the system}}$. If there are some odd terms τ in $prof(x)$, we can conjecture that he is the person to advertise his product or items. x has the similarity score of terms between $prof(x)$ and the blacklist *B* maintained by our spam filtering system.

III. Modeling Features for Bayesian Network Classifier

For applying our framework to the BNC, we separate feature values to states (i.e., categories) per feature. For instance, for the *request reject ratio* (*RR*) feature, we can categorize $0 \leq RR \leq 1$ to three states such as Low (L), Medium (M), and High (H). Suppose that we assign L = [0, 0.3], M = [0.3, 0.7], and H = [0.7, 1]. If *RR*(x) = 0.35, *RR*(x) = M. In the same way, we can model as follows:

- *Request Reject Ratio RR*(x) = L, M, or H.
- *Request Acceptance Ratio AR*(y) = L, M, or H.
- *Celebrity CB*(x) = L, M, or H.
- *Trust TR*(x) = L, M, or H.
- *Content Analysis CA*(x) = Y (Yes) or N (No).
- *Personality PS*(y) = E (Extrovert) or I (Introvert).
- *Friend's Friend FF*(x, y) = Y (Yes) or N (No).
- *Same Community SC*(x, y) = Y (Yes) or N (No).
- *Commonness CM*(x, y) = Y (Yes) or N (No).

A. Notation for Online Learning of Bayesian Networks

In this section, attributes A_i or A_j stands for features. From time to time, we call an attribute a (random) variable. As an example, A_i is *personality*. Each attribute contains a set of states $\{a_i^1, \ldots, a_i^{r_i}\}$ – e.g., a set of the personality feature is {*extrovert, introvert*}. C is a class label that contains one of two states c: *spam* and *non-spam*. $|A|$ is the number of attributes, and $|a|$ is the number of states.

B. Structure Learning

Structural learning is to find the best network that fits the available data. To construct the structure of the Bayesian network, we use *conditional mutual information* between attributes given the class variable [11] [12]. This function is defined as

$I_P(A_i; A_j|C) = \sum_{a_i,a_j,c} P(a_i, a_j, c)\log\frac{P(a_i,a_j|c)}{P(a_i|c)P(a_j|c)}$. This function measures the information that A_j provides about A_i when the value of C is known.

To accomplish the structural learning in the online Bayesian network learning, our proposed algorithm is also upgraded to have some online properties such as updating the network parameters and its structure adaptively. The outline of our algorithm can be given as follows:

1) Collect data.
2) Define the variables from the available data.
3) Start with a network with no links.
4) Compute $I_P(A_i; A_j|C)$ between each pair of attributes $(i \neq j)$.
5) Build a complete undirected graph in which the nodes are the attributes $A_1, \dots, A_{|A|}$. Annotate the weight of an edge connecting A_i to A_j by $I_P(A_i; A_j|C)$.
6) Build a maximum weighted spanning tree. The run time is $O(|A|^2\, log|A|)$, where $|A|$ is less than 10 in our framework.
7) Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.
8) Construct a tree structure by adding a node labeled by C and adding an arc from C to each A_i.
9) Update the network parameters along with new data.
10) Update the network structure:
    - If enough new data obtained, go to step 1) and generate a new network structure.
    - If no structural update is necessary, go to step 9).

C. Parameter Learning

The parameter learning has to calculate the values in the conditional probability table in a node (random variable). This task corresponds to Step 9) in the above structure learning. Let us assume that a fixed structure of the network. The learning is then the estimation of the *conditional probability table* (CPT) entries of the network.

Let A_i be a node (random variable) in the network that takes any value from the set $\{a_i^1, \dots, a_i^{r_i}\}$. Let Pa_i be the set of parents of A_i in the network that takes one of the configurations denoted by $\{Pa_i^1, \dots, Pa_i^{q_i}\}$. An entry in the CPT of the variable A_i is given by $\theta_{ijk} = P(A_i = a_i^k | Pa_i = pa_i^j)$. We are given a set of data cases $D = \{d_1, \dots, d_T, \dots\}$, and we have a current set of parameters θ that define the network. The data are either complete, that is all values of the variables are given, or incomplete.

The updating of the network parameters is achieved by the following maximization: $\tilde{\theta} = argmax_\theta[\eta L_D(\theta) - d(\theta, \bar{\theta})]$ where $L_D(\theta)$ is the normalized log likelihood of the data given the network, $d(\theta, \bar{\theta})$ is a distance (e.g., Chi squared distance) between the two models and η is the learning rate. Solving the maximization under the constraint that $\sum_k \theta_{ijk} = 1$ for $\forall_{i,j}$, Bauer et al. [13] derived an algorithm which they named $EM(\eta)$. Adapting the $EM(\eta)$ algorithm to the online learning case is as follows: The evidence becomes a single instance of the network and for each new evidence vector, the network's parameters are all updated according to the rule: $\tilde{\theta}_{ijk}^T =$

$$\begin{cases} \theta_{ijk}^{T-1} + \eta\left[\frac{P(a_i^k, Pa_i^k|d_T, \theta^{T-1})}{P(Pa_i^k|d_T, \theta^{T-1})} - \theta_{ijk}^{T-1}\right] \text{ for } P(Pa_i^k|d_T, \theta^{T-1} \neq 0) \\ \theta_{ijk}^{T-1} \text{ otherwise} \end{cases}$$

The online update method above is named the *Voting EM* algorithm [14]. Such online update rule is referred to as stochastic learning, with $\frac{P(a_i^k, Pa_i^k|d_T, \theta^{T-1})}{P(Pa_i^k|d_T, \theta^{T-1})}$ being the instantaneous gradient estimate of the constraint optimization problem. The learning rate η controls how much we rely on the past. As η approaches 1, the past is weighted less, and the update of the parameters is based more on the present data. As η approaches zero, the network parameters change slowly from the previous model. The learning rate can either be constant or adaptive.

D. Training and Testing for BNC

Suppose that the structure of the Bayesian network shown in Figure 1 is created in terms of the structure learning algorithm in Section *B*. In the figure, random variable C stands for the root variable, in which there are two cases: $C = spam$ and $C = non\_spam$. Figure 1 indicates the Bayesian network in case of $C = spam$. More specifically, RR is a random variable (feature) indicating Request Reject Ratio. As discussed above, RR has a set of three states {L, M, H}. In addition, each state (e.g., L) has the probability (relative frequencies) which appear in the training set.

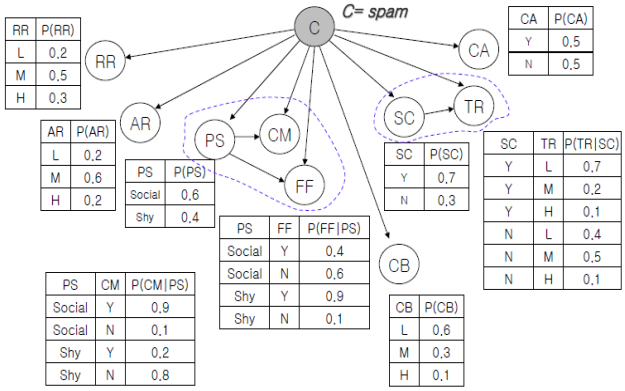

Figure 1. A Bayesian network with random variables (e.g., RR) and their corresponding probability distributions.

Note some random variables circled in the dotted line. For instance, the features are PS (*Personality*), CM (*Commonness*), and FF (*Friend's friend*). FF has the two parent variables – C and PS. We associate with FF a

conditional probability distribution, which specifies for FF the probability distribution over the values of FF given each combination of values for its parents – C and PS. Intuitively, there may be the correlation between FF (the child node) and PS (the parent node). If the personality of y is introvert, she is more likely to accept the friend-request of x in case that x is turned out to be her friend's friend.

Based on the Bayesian network in Figure 1, we become aware of the classification rule: $P(RR, AR, PS, CM, FF, CB, TR, SC, CA | C = spam) = P(RR | C = spam) \times P(AR | C = spam) \times P(PS | C = spam) \times P(CM | PS, C = spam) \times P(FF | PS, C = spam) \times P(CB | C = spam) \times P(SC | C = spam) \times P(TR | SC, C = spam) \times P(CA | C = spam)$ **(1.1)**.

In the training step, we can get equation (1.1) based on the training data set. Now our spam filtering system is keeping eye on a friend-request from x to y. Whenever x sends a friend-request to y, our spam filtering system simply computes values related to features. For instance, $RR(x) = \mathrm{M}$, $AR(y) = \mathrm{H}$, $PS(x) = \mathrm{E}$, $CM(x, y) = \mathrm{Y}$, $FF(x, y) = \mathrm{Y}$, $CB(x) = \mathrm{L}$, $SC(x, y) = \mathrm{Y}$, $TR(x) = \mathrm{M}$, and $CA(x) = \mathrm{N}$. Using equation (1.1) with probabilities, which are corresponding to the feature values in Figure 1, the system can determine whether the friend-request from x to y is a spam or not. More specifically, the classification formula is $Classify(A_1, \dots, A_n) = argmax P(C) \prod P(A_i | Pa(A_i), C)$, where A_i is i-th feature, C is spam or non-spam and $Pa(A_i)$ is the parent feature of A_i.

E. Likelihood Computation in BNC

The efficiency of BNC can be demonstrated by the computation of likelihood $P(C | A_1, \dots, A_{|A|}) = \frac{P(C)P(A_1, \dots, A_{|A|} | C)}{P(A_1, \dots, A_{|A|})}$ (i.e., posterior x likelihood = prior x likelihood / evidence).

When a user x sends a friend-request to the other user y, the likelihood of $P(A_1, \dots, A_{|A|} | C) = X \times Y \times Z$, where (1) $X = \prod_{i=1}^{|A|} P(A_i | P_a(A_i))$, where A_i and all parents of A_i depend only on x; (2) $Y = \prod_{j=1}^{|A|} P(A_j | P_a(A_j))$, where A_j and all parents of A_j depend only on y; and (3) $Z = \prod_{ij=1}^{|A|} P(A_{ij} | P_a(A_{ij}))$, where a_{ij} and all parents of A_{ij} depend on both x and y.

Then, X can be cached on x and Y can be cached on y. These quantities do not vary according to x or y. Only Z needs to be computed because it involves both sides.

From this likelihood computation technique, we can assure that our methods filter social spams is applicable for real world SNS domains. In addition, the algorithms that implement BNC can be very efficient even for very large data sets.

IV. Conclusion and Discussion

In this paper, we present a novel social spam detection framework, in which we extract and exploit effective features based on user behaviors and previous patterns of spammers. In addition, we devised incremental features and online classification model based on the personalized, multiple BNCs. In particular, our multiple classifiers work on the scalability problem. In conclusion, we can provide a guideline to manage the messages which looks like spams in SNSs.

For future works, the application in the real world will be needed to use this method more useful.

References

[1] Harris Interactive, "A study about Social Network Scams", 2008, "http://www.docstoc.com/docs/711319/socialnetworkingsurvey".

[2] S. Perez, "Social Networks and Spam", ReadWriteWeb, 2008, "http://www.readwriteweb.com/archives/social_networks_and_spam.php"

[3] D. Sculley and G. Wachman, "Relaxed Online SVMs for Spam Filtering", ACM SIGIR Special Interest Group on Information Retrieval (SIGIR'07), 2007.

[4] C. Castillo, D. Donato, and A. Gionis, "Know your Neighbors: Web Spam Detection using the Web Topology", ACM SIGIR Special Interest Group on Information Retrieval (SIGIR'07), 2007.

[5] M. Chang, W. Yih, and C. Meek, "Partitioned Logistic Regression for Spam Filtering", ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD'08), 2008.

[6] P. Kolari, A. Java, T. Finin, T. Oates, and A. Joshi, "Detecting Spam Blogs: A Machine Learning Approach", National Conference on Artificial Intelligenc (AAAI'06), 2006.

[7] B. Mehta, S. Nangia, and M. Gupta, "Detecting Image Spam using Visual Features and Near Duplicate Detection", International World Wide Web Conference(WWW'08), 2008.

[8] C. Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning, 1995.

[9] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer, "Learning Probabilistic Relational Models", 1999.

[10] L. Katz, "A New Status Index Derived from Sociometric Analysis", Psychometrika, 1953.

[11] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers", Kluwer Academic Publishers, Boston, 1997.

[12] F. Sahin, "A Bayesian Network Approach to the Self-organization and Learning in Intelligent Agents", Ph.D. dissertation, Virginia Polytechnic and State University, 2001.

[13] E. Bauer, D. Koller, and Y. Singer, "Update Rules for Parameter Estimation in Bayesian Networks", In Uncertainty in Artificial Intelligence (UAI), pages 3-13, 1997.

[14] I. Cohen, A. Bronstein, and F. Cozman, "Adaptive Online Learning of Bayesian Network Parameters", Technical Report, Hewlett-Packard Company 2001.

[15] R. Chen, K. Sivakumar, and H. Kargupta, "Collective Mining of Bayesian Networks from Distributed heterogeneous Data", Journal of Knowledge and Information Systems 2003.