Towards an ILP Approach for Learning Privacy Heuristics From Users' Regrets

Nicolás Emilio Díaz Ferreyra, Rene Meis, and Maritta Heisel

University of Duisburg Essen, Germany
{nicolas.diaz-ferreyra,rene.meis,maritta.heisel}@uni-due.de
https://www.ucsm.info/

Abstract. Disclosing private information in Social Network Sites (SNSs) often derives in unwanted incidents for the users (such as bad image, identity theft or unjustified discrimination), along with a feeling of regret and repentance. Regrettable online self-disclosure experiences can be seen as sources of privacy heuristics (best practices) that can help shaping better privacy awareness mechanisms. Considering deleted posts as an explicit manifestation of users' regrets, we propose an Inductive Logic Programming (ILP) approach for learning privacy heuristics. In this paper we introduce the motivating scenario and the theoretical foundations of this approach, and we provide an initial assessment towards its implementation.

Keywords: adaptive privacy, self-disclosure, awareness, social network sites, inductive logic programming

1 Introduction

Privacy norms play an important role in the construction of people's identity on an individual and collective level [3]. However, users of Social Network Sites (SNSs) like Facebook or Twitter seem to deliberately violate their privacy norms while interacting within these media platforms [2]. This is, users often share private or sensitive information that reaches unintended audiences in many cases. Consequently, users sharing acts derive in unwanted incidents (like stalking, identity theft, sextortion and job loss), along with a feeling of regret [4].

As one can see, there are two central aspects of information sharing: (i) not every information is appropriate to be mentioned in a particular context, and (ii) the audience to which information is disclosed influences such context. For instance, a user can consider acceptable to share his/her political affiliation with his family, but inappropriate with his/her work colleagues. Such aspects, which have been already identified in privacy theories (e.g. "contextual integrity" by Nissembaum [10]), are basic for the definition of privacy norms and consequently for individual privacy control.

In order to create a more privacy aware social environment, media technologies should support the users by providing guidance and feedback on their disclosures. Moreover, awareness mechanisms should be able to identify and learn

from regrettable experiences in order to provide effective privacy support. In line with these premises, Diaz Ferreyra et al. introduced the concept of Instructional Awareness System (IAS) which uses a privacy heuristics data base in order to generate adaptive awareness [5][4]. In this paper we propose to take a closer look into users' deleted posts and consider them as potential sources of privacy heuristics. Using Inductive Logic Programming (ILP) as the learning approach and deleted posts as the training set, we provide an initial assessment towards a privacy heuristics inference engine for IAS.

The content of this paper is organized as follows. In the next section, we discuss related work. Section 3 introduces the basic notions of ILP that form the paper's background. Following, in Section 4, we discuss how regrets can be identified and retrieved from users' deleted posts. In Section 5, we introduce a representation for regrettable scenarios and the initial components of an ILP system for learning privacy heuristics. Finally, we conclude in Section 6 with an outline and directions for future work.

2 Related Work

In the context of SNSs, adaptive privacy awareness systems seek to provide tailored feedback to the users when they attempt to reveal private information inside a post or in their profiles. Such feedback normally takes the form of a warning message that is displayed when a potentially unwanted disclosure is detected (e.g. a user revealing his/her bank account details in a public post on Facebook). For instance, Caliki et al. [2] developed a system which learns privacy rules from the user's previous sharing history, to use them later on as a criterion for raising awareness. This is, under the assumption that the user has never revealed private content to an unintended audience, the system infers *allow/deny(data, audience)* privacy rules though a machine learning engine (where "data" is a piece of private information like *bank account*, and "audience" is a sub-group of the user's Facebook friends). If a rule derived during the learning stage of the system is later violated in its operational stage, then a warning message is displayed. In line with this approach, Fang and LeFevre [6] introduced a system which analyses the information disclosed inside the user's Facebook profile in order to derive privacy preferences. This system recommends which attributes of the profile should be visible or not within a certain group of friends.

Following similar adaptivity principles, Díaz Ferreyra et al. developed an architecture which prescribes the basic components of an Instructional Awareness System (IAS) [5]. Using privacy best practices stored inside a Privacy Heuristics Data Base (PHDB), IAS generates a personalized warning message when a user attempts to reveal private data in a SNS post. The privacy heuristics in the PHDB are the outcome of a method which analyses the risks of regrettable online experiences reported by the users. This is, the input of the method are the experiences that users have reported themselves (to the development team of IAS for instance), or the outcome of an empirical research (e.g. questionnaires

or face to face interviews). This approach is effective for building a baseline of heuristics prior to the execution of the system. However, eliciting new entries of the PHDB requires the execution of this process which can be expensive and inefficient in terms of the resources and time needed to conduct interviews and process the outcome of them. Therefore, a *run-time* (*online*) approach for learning privacy heuristics would be beneficial for keeping up to date the content of IAS's PHDB.

3 Theoretical Background

This work proposes to endow IAS with a Privacy Heuristics Learning Engine (PHLE) based in principles of ILP. In this section we introduce the theoretical foundations of ILP and discuss its potential for carrying forward this task.

3.1 The ILP Problem

ILP is a discipline which employs techniques from machine learning and logic programming to infer hypotheses H from a set of observations and some background knowledge B [8]. Such observations are a set of positive and negative examples $E = E^+ \cup E^-$ of a concept to learn, and like the background knowledge, they are expressed as logic programs. The goal of ILP is to generate a logic program where positive examples are satisfied and negative examples are not. Let us consider that E^+ is expressed as ground unit definite clauses and E^- as ground unit headless Horn clauses of a single target predicate [7][1]. Then E contains ground unit clauses of a single predicate, and we can specify the general ILP problem as $B \wedge H \models E$. Since the goal is to find the simplest hypothesis, each clause in H should explain at least one example. If we consider H and E as single Horn clauses, then this expression can be rearranged as $B \wedge \neg E \models \neg H$.

Let $\neg\bot$ be the (potentially infinite) conjunction of ground literals which are true in all models of $B \wedge \neg E$ [9]. Considering that $\neg H$ must be true in every model of $B \wedge \neg E$, then it must contain a subset of the ground literals in $\neg\bot$. This is, $B \wedge \neg E \models \neg\bot$ and $\neg\bot \models \neg H$. Rearranging this expression, we obtain $H \models \bot$, which means that a subset of the solutions for H can be derived from $\bot$ by generalizing each given example in E. This means, $\bot$ is a clause that θ-subsumes[2] each example $e \in E$ [9]. This way, the final $\bot$ can be constructed as the disjunction of the body literals of all these derived clauses [9]. However, since $\bot$ can have infinite cardinality, the search space of those clauses which imply $\bot$ can also be infinite. In order to bound the search space of consistent and complete hypotheses, $\bot$ is built and generalized using *mode declarations* [9][11].

[1] Let $e(X)$ be the predicate which defines the examples, and $L = L_1, ..., L_n$ a set of ground literals which subsume the variable X. Then, positive examples can be expressed as $e(L_i)$., and negative examples as : $-e(L_j)$, $\forall\ 1 \leq i, j \leq n$

[2] A clause c_1 θ-subsumes a clause c_2 if and only if there exists a substitution θ such that $c_1\theta \subseteq c_2$. Consequently c_1 is a generalization of c_2 (and c_2 specialization of c_1) under θ-subsumption [8]

3.2 Mode Declarations and Types

Mode declarations are *language bias* (i.e. syntactic restrictions) that are imposed on candidate hypotheses in order to make the search space more efficient [9]. A mode declaration has either the form $modeh(n, s)$ or $modeb(n, s)$ for the head or body of a rule, respectively[9]. Consequently, it restricts the predicates which can occur in the head and body of a rule. The schema s is a ground literal with placemarkers of the form '*+type*', '*-type*', and '*#type*' standing for input variables, output variables, and constants, respectively [9]. The value n is the *recall* of the mode declarations, and is used to bound the number of times the scheme can be used. Let us consider the following example:

```
:- modeh(1, fly(+bird))?
:- modeb(1, wings(+bird,#property,-int))?
property(has_flight_feathers).
```

The first mode declaration says that general rules may have heads containing the predicate $fly(X)$ where X is a type of *bird* [1]. The second says that general rules may have bodies containing the predicate $wings(X, has\_flight\_feathers, Y)$, where X is a type of *bird*, Y an *integer*, and $has\_flight\_feathers$ a *property* [1]. The value of the recall here is one; therefore both mode declarations can be used once in the construction of the bottom clause.

4 Regrets Identification in SNSs

As discussed in the previous section, ILP is a supervised machine learning approach that can serve the purpose of developing a Privacy Heuristics Learning Engine (PHLE) for IAS. One sub-task in the development of such PHLE is to generate a set of examples (training set) to provide as input to the hypotheses derivation process. Since in many cases the information enclosed within a post has a private or sensitive semantics, posts become elemental units for user-centered privacy analysis in SNSs. In this sense, performing a privacy analysis of a post consists of determining the existence of private information enclosed in it. Such privacy analysis of user-generated content requires a taxonomy of attributes that can be considered as private. For this purpose, Díaz Ferreyra et al. proposed to organize the users' private or sensitive personal attributes around different high-level categories called the "self-disclosure dimensions" (i.e. demographics, sexual profile, political attitudes, religious beliefs, health factors and condition, location, administrative, contact, and sentiment) [4]. Each dimension consists of a set of Surveillance Attributes (SAs)[3] which allow to analyze from a user-centered perspective the information disclosed inside a post. For instance, in the scenario described in Fig. 1, the SAs *employment status*, *work location*, and *negative* sentiment are disclosed inside a post.

[3] SAs are those which can be linked to an individual, groups or communities and can raise privacy concerns related to data aggregation, probabilistic re-identification and undesirable social categorizations [4].

USER'S POST *"What a lame environment at the office...I can only hear people complain! Thanks God it's Friday! #tgif"*

Actual Audience: PUBLIC.
Unintended Audience: The user's work colleagues, or superior.
Unwanted Incidents: Wake-up call from superior; bad image; job loss.

Fig. 1. Example of self-disclosure scenario

When one or more SAs reach an unintended audience, an unwanted incident can take place and derive in a feeling of regret. In Fig. 1, the post reached the user's work acquaintances causing a wake-up call from superior, together with a negative impact on the user's image, and eventual job loss. Since the likelihood and impact of such incidents define the risks of the given scenario, they can be used to model a regret. This is, regrets can be expressed in terms of the SAs, unintended audience, and the associated risks (i.e. likelihood and impact of an unwanted incident). Risks, likelihood and consequence, can be expressed in a nominal scale. In this sense, a consequence is a value on an impact scale such as insignificant, minor, moderate, major or catastrophic. Likewise, the likelihood is a value on a frequency scale such as rare, unlikely, possible, likely and certain. Finally, a risk is a value obtained from the likelihood and consequence of the unwanted incident and expressed in a scale such as very low, low, high and very high.

5 Towards the development of a Privacy Heuristics Learning Engine

So far we have discussed the concepts that serve in the identification of regrettable posts and therefore for creating examples for a PHLE. However, in practice, when a post is deleted there is not much information about the risks and, moreover, if the deleted post has derived or not in a regret for the user. In this section we provide an approach on how this missing information can be retrieved and used later on for the development of privacy heuristics. We also introduce a syntax for such heuristics in the IAS's PHDB together with the respective mode declarations for their automatic inference.

5.1 Privacy Heuristics

Snippet 1 describes a syntax for the entries in the PHDB, namely the hypotheses (i.e. regret conditions) to be learned by the PHLE. These learned hypotheses are the privacy heuristics which will help to identify potential regrettable scenarios in the future. That is, when a user attempts to disclose SAs inside a post, IAS should query the PHDB in order to verify that this will not lead to a potential regret. In other words, we want to find the consequence (Cons) and

frequency (Freq) of a potential unwanted incident (Unwi) based in the disclosed SAs ([X|Xs]) and the list of users that conform the post's audience ([Y|Ys]). The *regret* predicate models this conceptual relation, and is the entry point for querying the PHDB. Consequently, [X|Xs] and [Y|Ys] are input variables of a *regret-?* query, and Unwi, Cons and Freq are the output variables.

The impact of an unwanted incident (and consequently the acceptance level of the risk associated with it) are perceived differently among individuals. For instance, some users might consider the consequence associated with a "wake up call from supervisor" as Insignificant, and others as Catastrophic. This is because individuals have different privacy attitudes which influence the perception level of risky events [4]. In order to model this, we will adopt the Westin's Privacy Index for the classification of the users' privacy attitudes [12]. Such index classify individuals into three privacy groups: fundamentalists, pragmatists, and unconcerned (each group with high, medium and low levels of privacy concerns respectively). The predicate *acceptable*, represents this relation between the user's privacy attitude and the risk acceptance level associated with it.

```
/*Privacy Heuristic*/
regret([X|Xs], [Y|Ys], Unwi, Cons, Freq):-
  srv_att_list([X|Xs]), usr_list([Y|Ys]), inSIG([Y|Ys], SIG1),
  unwanted_inc([emp_status, work_location, neg], SIG1, wake_up_call),
  subset([X|Xs], [emp_status, work_location, neg]),
  risk(wake_up_call, Cons, Freq, Level), not acceptable(Att, Level).
```

Snippet 1. Example of a Privacy Heuristic

Another element of a regret is the audience towards which a set of SAs has been disclosed. Normally in SNSs like Facebook, a user has a list of "friends" composed by other users of Facebook. We will adopt this approach and assume that the user's friends list is grouped into different circles which are constructed under the premises of Social Identity Theory (SIT). Basically SIT postulates that people belong to multiple social identities (for instance, being Swedish, being an athlete, or being a researcher are all examples of identities/identity groups). Since users frequently have a mental conceptualization of the different social identity groups with whom they interact, we will assume that the friends list of a user is clustered into a set of identity groups as suggested by SIT. We can imagine for instance groups like *work*, *church*, *gym*, or *choir* depending on the social circles that a user belongs to. The predicate *inSIG*, evaluates if the audience to which the post has been disclosed corresponds to one of the social identity groups (SIG) in which the user's friends list is clustered.

To learn the circumstances of a regrettable scenario means to define instances of some of the variables which appear in the body of the *regret* predicate. Particularly, we are interested in knowing which concrete SAs lead to a certain unwanted incident when disclosed to a particular SIG. Consequently, the SAs, unwanted incident and SIG involved in the scenario of Fig. 1, are represented by the ground literals *[emp_status, work_location, neg]*, *wake_up_call*, and *SIG1* re-

spectively (we will consider *SIG1* as the identifier of the SIG "work place"). The relation between these elements is modeled by the *unwanted_inc* predicate, and the *subset* predicate verifies if the SAs involved in the unwanted incident are a subset of the SAs disclosed inside the deleted post. Likewise, the *risk* predicate models the semantic relation between the unwanted incident (*wake_up_call* in this case), its frequency and its consequence (as described in Section 4). As already mentioned, the risk of an unwanted incident (which is expressed as a level on a risk scale) can be acceptable or nor depending on the user's privacy attitude. For instance, we can assume that for fundamentalists only very low risks are acceptable, for a pragmatist very low and low risks, and for an unconcerned the risks which are very low, low and high. The acceptance of the risk level, will at the end determinate if the disclosure scenario which is being evaluated will lead to a potential state of regret or not.

5.2 Regrets Retrieval

Once the relevant SAs are identified, the information about the causes which motivated the user to delete the post have to be retrieved. Since this information is not evident at a glance, we propose to build an interface which asks questions to the user when a "delete" event takes place. That is, when a post is deleted, we first analyze if it contains SAs, and if so we ask the user for extra information to complete the description of the regrettable scenario. A mock-up of the described interface is depicted in Fig. 2, where the reported risk information corresponds to the post described in Fig. 1.

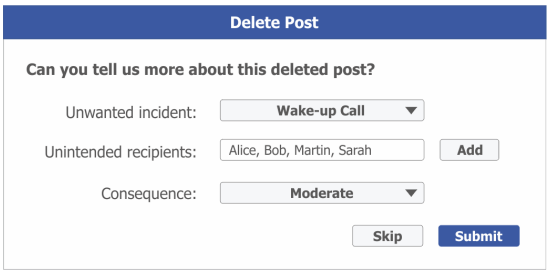

Fig. 2. "Delete Post" Interface

Among the information requested from the user there is the "unintended recipients" of the post. This corresponds to a list of users which were part of the original audience of the post, but should have not been included for privacy reasons. In the example, the user reports that the post reached his/her work colleagues Alice, Bob, Martin and Sarah, and has selected them as the unintended recipients. Likewise, the user reports that the unwanted incident has been a *wake-up call* and that the consequence has been perceived by him/her as *moderate*. With this information submitted by the user (i.e. unintended recipients, unwanted incident, and consequence), and the list of SAs extracted

from the post, we can create a *regret* predicate consisting of the ground literals: *[emp_status, work_location, neg]*, *[Alice, Bob, Martin, Sarah]*, *wake_up_call*, *moderate*, *certain*. This *regret* predicate represents a concrete regrettable scenario, and is therefore a training example for the PHLE.

Once the regret scenario is submitted by the user, IAS will first store the information about the risks. That is, it will generate a *risk* entry in the PHDB with the information about the unwanted incident and its consequence. Since a *risk* predicate contains also the frequency of the unwanted incident and the risk level, such values must be also generated. For the frequency, we will adopt "certain" for every case since we can assume that if a post containing SAs has been deleted is because a regret took place. Likewise, risks levels can be defined for every value combination of likelihood and consequence. We will assume that when the likelihood is certain and the consequence is moderate, the risk level is very high. Consequently, for our example, a *risk* predicate with the ground literals *wake_up_call*, *moderate*, *certain*, and *very_high* is created as a new entry in the PHDB. Risks together with the information of their acceptance level (the *acceptable* predicates), are part of the background knowledge of the PHDB. In this case we will consider that the user is a *pragmatist*, and therefore only risks which are *low* or *very_low* are acceptable (see Snippet 2).

```
/*Regret Example*/
regret([emp_status, work_location, neg], [Alice, Bob, Martin, Sarah],
    wake_up_call, moderate, certain).
/*Background Knowledge*/
risk(wake_up_call, moderate, certain, very_high).
sig([Bill, Bob, Sam, Sarah, Alice, John, Martin], SIG1).
acceptable(pragmatist, low).
acceptable(pragmatist, very_low).
```

Snippet 2. A regret example submitted by the user

Another element which is part of the background knowledge are the different SIGs in which the user's friend list is grouped. As can be observed in Snippet 2, the predicate *sig* assigns an identifier to each of the SIGs. For instance, SIG1 refers to a group of users consisting of *[Bill, Bob, Sam, Sarah, Alice, John, Martin]*. This information allows to derive heuristics where the audience is generalized to a SIG. This is, based on the examples of regrettable scenarios, identify the SIG to which a set of SAs should not be revealed to. In Snippet 2, we can see that the unintended recipients could be generalized to SIG1. Consequently, the next time the user attempts to disclose the SAs *[emp_status, work_location, neg]* to an audience containing a member of SIG1, then IAS will raise a warning message.

5.3 Mode Declarations and Type Definitions

In order to define the mode declarations, first it is necessary to define the *types* of the elements which are part of them. That, is to describe the categories of

objects (number, lists, names, etc) in the domain being modeled. In our case we need the types *Cons*, *Freq*, *Unwi*, *Usr*, *SA*, *Att*, *Level*, *srv_att_list*, *usr_list*. Due to space limitations we only provide a partial description of these in Snippet 3. The missing types can be defined in an analogous way. Once these types are defined, we can then proceed with the definition of the head and body mode declarations. The head of a heuristic (i.e. a *regret* predicate) is a function of objects of the types *srv_att_list*, *usr_list*, *Unwi*, *Cons*, and *Freq*. All of these objects are variables in the head of the *regret* predicate, however (as mentioned in Section 5.1), only *srv_att_list* and *usr_list* are input variables. Therefore, the placemarkers of the *modeh* for the head of the *regret* predicate will be *+srv_att_list*, *+usr_list*, *-Unwi*, *-Cons*, and *-Freq*.

```
/*Types*/
SA(emp_status). SA(work_location). SA(neg).
...
srv_att_list([]).
srv_att_list([X|Xs]):-SA(X), srv_att_list(Xs).
/*Mode declarations*/
:- modeh(1, regret(+srv_att_list, +usr_list, -Unwi, -Cons, -Freq))?
:- modeb(1, inSIG,(+usr_list, #SIG)?
:- modeb(1, unwanted_inc(#srv_att_list, #SIG, #Unwi))?
:- modeb(1, subset(+srv_att_list, #srv_att_list)?
:- modeb(1, risk(#Unwi, -Cons, -Freq, -Level))?
:- modeb(1, not acceptable(+Att, +Level))?
```

Snippet 3. Types and mode declarations

The body of the *regret* predicate is defined by the predicates *inSIG*, *unwanted_inc*, *subset*, *risk*, and *not acceptable*. Therefore, we need to define a body mode declaration *modeb* for each one of these predicates. Following the same criterion used to define the input and output variables of *modeh*, we define the arguments for each *modeb*. Since many of the predicates used in the body of *regret* are facts or contain grounded literals in their declaration, we express their respective *modeb* using the *#type* placemarker. Such is the case of the *risk* clause which in the body of the *regret* rule has its parameter *Unwi* defined as a constant. Likewise, the argument *SIG* in *inSIG*, *srv_att_list* in *subset*, and all the arguments of *unwanted_inc* are grounded literals in the body of *regret*. Therefore, we express them as constants in their respective body mode declarations.

6 Conclusion and Future Work

So far, we have introduced an ILP model of a PHLE which learns privacy heuristics from regrettable posts in SNSs. We believe that this is a promising approach which will contribute in shaping better and more user-centered preventative technologies. It is a matter of future research to develop a prototype of the model here introduced in order to measure its performance, as to evaluate if adaptive

awareness approaches like IAS indeed help to achieve better engagement levels. Other directions for future research involve the development of alternative approaches to Westin's Privacy Index for measuring the users' perceived severity of unwanted incidents (as suggested by Díaz Ferreyra et al. [4]). This involves the development of a user interface (in the form of a short questionnaire) to capture the users' attitudes towards privacy risks in order to improve the performance of the PHDB and consequently of IAS.

Acknowledgments. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media".

References

1. Athakravi, D., Broda, K., Russo, A.: Predicate invention in inductive logic programming. In: OASIcs-OpenAccess Series in Informatics. vol. 28. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2012)
2. Calikli, G., Law, M., Bandara, A.K., Russo, A., Dickens, L., Price, B.A., Stuart, A., Levine, M., Nuseibeh, B.: Privacy Dynamics: Learning Privacy Norms for Social Software. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 47–56. ACM (May 2016)
3. Diaz, C., Gürses, S.: Understanding the landscape of privacy technologies (extended abstract). In: Proceedings of the Information Security Summit, ISS 2012. pp. 58–63 (May 2012)
4. Díaz Ferreyra, N.E., Meis, R., Heisel, M.: Online Self-disclosure: From Users' Regrets to Instructional Awareness. In: Proceedings of the IFIP International Cross-Domain Conference (CD-MAKE) (August 2017), Accapted for publication
5. Díaz Ferreyra, N.E., Schäwel, J., Heisel, M., Meske, C.: Addressing Self-disclosure in Social Media: An Instructional Awareness Approach. In: Proceedings of the 2nd ACS/IEEE International Workshop on Online Social Networks Technologies (OSNT). ACS/IEEE (December 2016)
6. Fang, L., LeFevre, K.: Privacy wizards for social networking sites. In: Proceedings of the 19th International Conference on World Wide Web. pp. 351–360. WWW '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1772690.1772727
7. Muggleton, S.: Inverse Entailment and Progol. New Generation Computing 13(3), 245–286 (1995)
8. Muggleton, S., De Raedt, L.: Inductive Logic Programming: Theory and Methods. The Journal of Logic Programming 19, 629–679 (1994)
9. Muggleton, S.H., Firth, J.: CProgol4.4: a tutorial introduction. In: Dzeroski, S., Lavrac, N. (eds.) Relational Data Mining, pp. 160–188. Springer-Verlag (2001), http://www.doc.ic.ac.uk/\~shm/Papers/progtuttheo.pdf
10. Nissenbaum, H.: Privacy as contextual integrity. Wash. L. Rev. 79, 119 (2004)
11. Roberts, S.: An introduction to progol. Department of Computer Science, University of York 244 (1997)
12. Westin, A.F.: Privacy and Freedom. Washington and Lee Law Review 25(1), 166 (January 1968)