A Privacy Preserving Web Recommender System

Ranieri Baraglia
HPC Lab. ISTI-CNR
Pisa, Italy

Claudio Lucchese
Universita’ Ca’ Foscari
Venezia, Italy

Salvatore Orlando
Universita’ Ca’ Foscari
Venezia, Italy

Massimo Serrano*
HPC Lab. ISTI-CNR
Pisa, Italy

Fabrizio Silvestri
HPC Lab. ISTI-CNR
Pisa, Italy

ABSTRACT
In this paper we propose a recommender system that helps users to navigate through the Web by providing dynamically generated links to pages that have not yet been visited and are of potential interest. To this end, traditional recommender systems use Web Usage Mining (WUM) techniques in order to automatically extract knowledge from Web usage data. Thanks to WUM techniques we are able to classify users and adaptively provide useful recommendations. The drawback of a user classification approach is that it makes the system prone to privacy breaches.

Our contribution here is πSUGGEST, a privacy enhanced recommender system that allows for creating serendipity recommendations without breaching users privacy. We will show that our system does not provide malicious users with any mean to track or detect users activity or preferences.

Categories and Subject Descriptors
K.4.1 [Public Policy Issues]: Privacy; I.5.1 [Models]: Statistical

General Terms
Algorithms, Security

Keywords
Web Recommender Systems, Privacy Preserving User Modeling

1. INTRODUCTION
The continuous and rapid growth of the Web has led to the development of new methods and tools in the Web recommender or personalization domain. In [6] the goal of the Web personalization is defined as “provide users with the information they want or need, without expecting from them to ask for it explicitly”.

Web Mining has shown to be a viable technique to discover information “hidden” into Web-related data [4]. In particular, Web Usage Mining (WUM) is the process of extracting knowledge from Web users access data (or click-stream) by exploiting Data Mining (DM) technologies [5]. It can be used for different purposes such as personalization, system improvement and site modification.

The knowledge extracted is actually a classification model for users in different groups with different interests. Obviously the presence of such classification system can introduce privacy breaches, by either disclosing personal information or allowing malicious queries capable of reconstructing the knowledge collected by the system. In this work we mainly focus on this last aspect, and present a WUM system, called πSUGGEST, which is designed to dynamically generate personalized contents of potential interest for users of a Web Site, without providing any privacy breaches to malicious users. The architecture of πSUGGEST is based on a two-tier structure. One of them has to be plugged-in into the browser at the client-side. The other tier is based on an incremental personalization procedure, tightly coupled with the Web server. Its knowledge base is incrementally updated by monitoring usage data, and then notified to the client, which will be able to use it to personalize on-the-fly the requested HTML page, by appending a list of page links (suggestions).

Eventually, we define a measure of privacy in order to evaluate with which confidence a malicious user can infer users activities from the provided suggestions. The quality of suggestions was evaluated by adopting the metric introduced in [2]. This metric tries to estimate the effectiveness of a recommendation system as the capacity of anticipating users requests that will be made farther in the future.

Summarizing, the main contributions of this work are: an algorithm to incrementally generate users profiles in a privacy preserving way. A general privacy measure for classification - based Web recommender systems. Finally we will show that πSUGGEST successfully preserves users privacy w.r.t. the measure we introduced.

The rest of the paper is organized as follow. In Section 2 we show some works related to this paper. Section 3 presents the architecture and the algorithms used by πSUGGEST. Section 4 presents a framework for analyzing privacy in cluster-based recommender systems in general and we adopt it for the analysis of πSUGGEST’s privacy. Finally in Section 5 we conclude the paper by presenting some forthcoming work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC ’06 April 23-27, 2006, Dijon, France
Copyright 2006 ACM 1-59593-108-2/06/0004 ...$5.00.
privacy breach, where by privacy breach we mean the chance
of the ratings expressed by similar users, but that have not
user enters the system, the given recommendations consists
therefore to detect different habits among users. Once a new
social network over users data can be built by linking sim-
mock distance

i.e., by a statistically relevant group. We could think that
actual behavior of a group of users.

2. PRIVACY PRESERVING WEB RECOM-

MENDER SYSTEMS

In the past, several WUM projects have been proposed to
foresee users preference and their navigation behavior. In
the following we review some of the most significant WUM
projects that can be compared with our system. The survey
in [1] contains an overview of these systems.

To the best of our knowledge, no Web Recommendation
Systems that take into account privacy concerns have been
yet designed. Only a preliminary work [7], aimed to formal-
ize the problem and give a measure of the amount of privacy
provided to the users, has been presented. In that paper a
recommendation system is seen as a user classifier. Users
who share at least \( w \) identical ratings (e.g. they visited the
same \( w \) web pages) can be considered similar, with ham-
mock distance \( l \) = 1. Given this similarity relationship, a
social network over users data can be built by linking sim-
ilar users. This network will be likely to form groups and
therefore to detect different habits among users. Once a new
user enters the system, the given recommendations consists
of the ratings expressed by similar users, but that have not
been already expressed by the user himself.

Classifying users in such a way turns out to be a strong
privacy breach, where by privacy breach we mean the chance
for a malicious user to track users activities or preferences.
For example, suppose that a user rates items \( \{a, b, c, d\} \) (e.g.
items can be web pages) and receives as a recommendation
item \( e \). Then we know that there is a bunch of users who
actually rated all the items \( \{a, b, c, d, e\} \) at the same time.
This is a first kind of breach, since we have detected the
actual behavior of a group of users.

Moreover, recommendations are usually given only when
they are supported by a certain number \( \text{minfreq} \) of users,
i.e., by a statistically relevant group. We could think that
if just a single user has rated items \( \{a, b, c, d, e\} \), since this
information will not be considered during the classifier train-
ing, his privacy will be preserved. However, a malicious user
could perform consecutive interactions with the system and
discover that after rating \( \{a, b, c, d, e\} \) for \( \text{minfreq} \) – 1 times,
this new pattern will appear in the recommendations, thus
detecting the preferences of one single user. In other words,
such a system can be exposed to queries and this is a second
kind of privacy breach.

3. THE \( \pi \text{SUGGEST} \) SYSTEM

\( \pi \text{SUGGEST} \) is an evolution of \text{SUGGEST} with a strong
difference in its architecture. The two components aiming
at updating the knowledge base, and creating recommen-
dations are separated (see Figure 1). The first is placed on
the web server implemented as a module of the Apache Web
Server. The second one works on the client-side as a browser
plugin.

In order to catch information about navigational patterns,
\( \pi \text{SUGGEST} \) does not need to maintain in a database the
complete sessions associated with the various users of the
Web site. On the other hand, it only needs to maintain
a particular data structure, i.e. an undirected graph with
weighted edges \( G = (V, E) \), from which recommendations
can be extracted. In particular, the set \( V \) of vertices con-
tains the identifiers of the different pages hosted on the Web
site. Based on the fact that the interest in a page depends
on its content and not on the order a page is visited during
a session, we assign to each edge \( E \) a weight computed as:
\( W_{ij} = N_{ij}/\max\{N_i, N_j\} \), where \( N_{ij} \) is the number of
times pages \( i \) and \( j \) have been accessed consecutively and in
any order by a user, while \( N_i \) and \( N_j \) are respectively the
number of times page \( i \) or page \( j \) has been visited. Note
that the sessions are not materialized and values are up-
dated entirely on-line. Dividing by the maximum between
single occurrences of the two pages has the effect of reduc-
the relative importance of links involving index pages.
Such pages generally do not contain useful content and are
used only as a starting point for a browsing session. More-
over, often users return back to such pages several times,
in order to start the visit of a new branch of the Web site.
Therefore, index pages are very likely to be visited with any
other page and nevertheless are of little interest as potential
suggestions.

A triangular adjacency matrix \( N \) is used to store the
knowledge base, where each entry \( N[i,j] \) contains the value
\( N_{ij} \). We assume that each entry \( N[i,i] \) stores value \( N_i \).
The adjacency matrix is incrementally maintained, by only
considering single HTTP requests coming from clients. Each
request consists of the identifier of the requested page, along
with the identifier of the page which the user is coming from.
The last page is the referral one, which is usually transmit-
ted together with each HTTP request (see Figure 1).

The \( \pi \text{SUGGEST} \) component on the server, besides main-
taining the adjacent matrix \( N \) modeling the undirected wei-
ighted graph \( G \), also finds disjoint clusters of strongly cor-
related pages by partitioning \( G \) on the basis of its con-
ected components. To this end, \( \pi \text{SUGGEST} \) actually uses a
modified version [2, 3] of the well known incremental con-
ected components algorithm [8]. Moreover the information
about the cluster identifiers, associated with the various ver-
tices of \( G \), is maintained in another data structure, a vector
\( L \). In large Web Sites the size of the adjacency matrix \( N \)
and the vector \( L \) might exceed the maximum available main
memory. We thus adopted a LRU-based strategy to store
in main memory portions of the data structures associated
with those pages that have been recently accessed by some
users.

\( \pi \text{SUGGEST} \) works as follows. On the client-side, when
a session starts, the plugin asks the server for page clus-
ters (stored in \( L \)) extracted from the knowledge base. At
the same time, the plugin is also responsible for tracking
the user and holding her/his session. The plugin also cre-
ates the suggestions for the user. Suggestions are built in
a straightforward manner by finding the cluster that has
the largest intersection with the PageWindow related to the
current session. The final suggestions will include the most
relevant pages in the cluster, according to an order determined by the clustering phase.

Note that session information is never disclosed by the \( \pi \text{SUGGEST} \) client, since such information is not needed by the server to update its knowledge base. On the other hand, users sessions are exploited by other recommender systems in order to classify a user according to her/his behavior. This is a weak point of such systems from the point of view of privacy, since the information enclosed within a session can be listened by third party during the communication from the client to the server, or misused at the server side.

Our connected component algorithm, used on the server side to incrementally cluster Web pages on the basis of graph \( G \), is driven by two threshold parameters. The aim of these thresholds is to limit the number of edges to visit, but also to avoid the generation of clusters that over-fit the knowledge base and are not statistically relevant. In particular,

1. we filter from \( G \) the edges whose weight \( W_{i,j} \) is below a constant value, called \( \minfreq \). Elements \( W_{i,j} \) of \( W \) (i.e. links between pair of pages) whose values are less than \( \minfreq \) are poorly correlated and thus not considered by clustering algorithm;

2. we only consider connected components of size greater than a fixed number of nodes, namely \( \minclustersize \). All the components having less than \( \minclustersize \) nodes are discarded because considered not significant enough.

The evaluation of performance and effectiveness of our recommender system can be found in [2], where we also introduced a new effectiveness parameter, based on the intersection of real sessions with the corresponding set of suggestions. The test were conducted by using three real life access log files of public domain:\(^1\) : Berkeley, NASA, USASK.

4. \( \pi \text{SUGGEST} \) AND PRIVACY

In order to evaluate the privacy given by \( \pi \text{SUGGEST} \), we want to quantify with which level of confidence we can infer information about users activities.

In general, a recommender system tries to classify a user according to the pages s/he visited. Each class of users is associated with a subset of pages which are supposed to be interesting for them. In \( \pi \text{SUGGEST} \), the pages associated with each class are a public information, since the content of data structure \( L \) is returned to each client when a user session starts. In other systems we can assume that classes are maintained private, even if part of such information must be published in the form of user recommendations. Moreover, as we have seen previously, such classes can be inferred with a kind of query-driven interaction. In the following we will always refer to a class as a cluster of pages, and we will investigate which kind of information is revealed together with the information relative to the composition of a generic cluster.

From the point of view of the plugin on the client-side, a cluster simply is a set of pages \( C = \{p_0, p_1, ..., p_n\} \), even if \( C \) has been obtained by partitioning graph \( G \), and thus \( C \) actually corresponds to a (partially or completely) connected component of \( G \). However, the plugin on the client-side can not be aware of which pairs of pages actually correspond to edges that belong to the connected graph component behind \( C \). On the other hand, a user activity corresponds to a set of pages the user visited (with cardinality greater than 1). The user also moved from a page to another, and thus there must exist a partially or completely connected graph behind such set of pages.

We are interested in which kind of user activities may have generated a given cluster. To this end we introduce the concept of valid cluster generator.

In the rest of the paper the two parameters \( \minfreq \) and \( \minclustersize \) will not be considered anymore. In fact, they only affect the quality of the classification structure of \( \pi \text{SUGGEST} \) and not the theoretical results we are going to present.

**Definition 1.** Given a cluster \( C = \{p_0, p_1, ..., p_n\} \), and a set of user activities \( U = \{U_1, ..., U_m\} \), where each \( U_i \) is a subset of the pages that belong to \( C \) and have been visited by some user, \( U \) is a valid cluster generator if and only if the following three conditions hold:

1. covering \( \bigcup_{i=1}^{m} U_i = C \).
2. connectivity \( \forall U_i \in U, \exists U_j \in U, i \neq j, s.t. U_i \cap U_j \neq \emptyset \).
3. minimality \( \forall \{U \setminus U_i\} \text{ is not a valid cluster generator.} \)

Note that, since a connected graph exists behind each \( U_i \), the connectivity condition ensures that the union of all the connected graphs associated with the various \( U_i \) surely generates one of the possible connected graphs that are able to support/generate \( C \).

A cluster generator is simply a set of user activities (sessions), and it is valid if it is able to create the connected component \( C \) and if it is minimal. We introduce minimality to avoid anomalous combinations that may be useless in this context, e.g., we do not want \( \{\{a\}, \{b\}\} \) to be a valid generator for cluster \( \{abc\} \), since the cluster is also supported by the first user session only.

Different recommendation systems have different kinds of valid cluster generators. Nevertheless this concept is applicable to any of them.

**Definition 2.** Given a cluster \( C = \{p_0, p_1, ..., p_n\} \), and a valid cluster generator \( U \), the privacy level \( \Pi \) provided by a recommender system \( \Sigma \) w.r.t. \( U \) is the conditional probability:

\[ \Pi_2(U, C) = 1 - P(U \mid C) \]

The rationale is clear: if given a cluster \( C \), we can estimate \( U \) with high probability, the system provides very low privacy. On the other hand, if there is no \( U \) which is likely to happen with high probability, then the system provides a high level of privacy.

In Table 1, we give a small example to better understand the problem. Suppose that we receive from \( \pi \text{SUGGEST} \) one single cluster \( C = \{a, b, c, d, e\} \). We can figure out many different events that may have generated \( C \). For example, one single user may have visited all the pages \( \{a, b, c, d, e\} \), or two users may have visited respectively the pages \( \{a, b, c\} \) and \( \{c, d, e\} \), or three users may have visited the pages \( \{a, b, c\} \), \( \{a, c, d\} \) and \( \{d, e\} \), and so on. Note that different users activities, may generate not only the same cluster, but also the same internal representation of the knowledge base.

\(^1\)www.web-caching.com
valid cluster generators

\[ \exists \text{ item in } \{S\} \]

Firstly, we consider the case \(|\{S\}| = 1\), then we can take any subset \(C\) \(\cup\) \(\{b, c\}\) and thus no privacy. For the case \(|\{S\}| = 2\), we have only four valid cluster generators (see Table 1) leading to a privacy level of \(1 - \frac{1}{4}\).

According to our previous considerations, we know that \(|\{S\}| \geq 4\), thus we can rewrite the above formula with:

\[ |U| = \sum_{i=1}^{i=\infty-N-2} \left( \frac{|C|}{i} \right) \frac{2^{C-i}}{2} = \frac{2^{|\{S\}|} - 2}{2} \]

which means that: \(P(U \mid C) \leq \frac{1}{2^{|\{S\}|}}\). 

As expected, the amount of possible valid cluster generators is very high, and therefore it is not possible to understand which set of user activities have actually lead to cluster \(C\). But we are pretty much interested, not only in giving a confidence level for a set of users activities as above, but also a confidence level for the activity of a single user.

Definition 3. Given a cluster \(C = \{p_0, p_1, \ldots, p_n\}\), and a set of pages visited by a single user \(U = \{q_0, q_1, \ldots, q_{N_2}\}\) with \(U \subseteq C\), the privacy level \(\Pi^*\) provided by some recommender system \(\Sigma\) w.r.t. \(U\) is the conditional probability:

\[ \Pi^*_\Sigma(U, C) = 1 - P(U \mid C) \]

Proof. In the following we will show that, for each valid cluster generator \(U\) of \(C\), where \(U\) is included in \(U\), there are at least \(3 \frac{2^{|\{S\}|}}{2}\) valid cluster generators that do not include \(U\).
By definition, \( U \setminus U \) is not a valid generator. This can be due to covering or connectivity properties. Suppose that it is only due to connectivity. This means that the graph behind \( U' = (U_1, ..., U_n) \) is disconnected. This means that we can partition the graph by considering its connected components. Let \( \mathcal{G}_i \) be a connected component of that graph and let \( U_{\mathcal{G}_i} \) be the union of all the user activities covered by \( \mathcal{G}_i \). We have that

\[
\bigcup_{U_i \in U} U_i \setminus \bigcup_{U_i \in U} U_i = \emptyset
\]

Note that, for each group of activities \( \mathcal{G}_i \), we thus have a disjoint group of pages \( P_i \). To connect all these \( P_i \), we need at least a single new user activity \( U' \) that replaces \( U \), where \( \forall P_i, \exists \pi \in P_i \) s.t. \( p \in U' \). We have multiple possible ways to choose \( U' \) according to the above properties. Note that, when the new \( U' \) is selected, it is possible that we find some user activity \( U_i \in U \), such that \( U_i \subseteq U' \); in this case, due to the minimality property, \( U_i \) must be removed from the new cluster generator \( U' \).

The case that lower bounds the possible choices of \( U' \) is when each activities \( U_i \in U \) is made up of a single pair of pages, and the various \( U_i \) are disjoint. In this case we have that the possible way of choosing \( U' \) are

\[
(2^2 - 1) \cdot \frac{|C|}{|C|} = 3 \cdot \frac{|C|}{|C|}
\]

where \( (2^2 - 1) \) is the number of possible subsets of a set of two elements, without considering the empty set. Therefore we have that \( P(U \mid C) \leq \frac{1}{3^{|C|}} \).

Finally, if we also consider that \( U \setminus U \) is not valid due to the covering property, the possible choices of the substitute of \( U \) increase, so that the bound for \( P(U \mid C) \) still hold.

Theorem 1 and Theorem 2 lead us to the following conclusion. We state that if the \( \piSUGGEST \) system is plugged into a privacy safe system, it will not provide any privacy breach. We say that a system is privacy safe if the two conditions hold: (i) the user activity cannot be tracked, (ii) the user activity cannot be inferred. Condition (i) has to hold by definition in a safe system. If we add \( \piSUGGEST \) to such a system, the only additional parameter we would need is the current page. Since this parameter cannot discriminate a user among the others, it turns out to be impossible to use it to track users activity (e.g. listening the communication channel), and therefore we have that condition (i) still holds. Finally, neither publishing the clustered structure can be considered a privacy breach (however it could be inferred with consecutive queries to the system). Theorem 1 assures that the privacy provided by \( \piSUGGEST \) increases exponentially with the size of the published cluster. Given one recommendation, there are exponential many aggregate behavior that might have generated it, and therefore it is not possible to detect the actual behavior among them, i.e. condition (ii) holds.

5. CONCLUSION

In this work, we presented a privacy enhanced web recommender system. State-of-the-art algorithms require users to be classified in order to provide them with interesting suggestions. This classification-based approach has been shown to be a privacy breach itself. It reveals which pages a group of users have actually visited. This information may be used by potential competitors to, for instance, restructure their own sites according to the usage patterns “stolen” from a site which uses privacy-disclosing Web recommender systems. According to this framework of user classification based systems, we define a new privacy measure. This metric models the chance for a malicious user to recover the real behavior of a group or a single user, on the basis of the information revealed by the system. Finally we introduced \( \piSUGGEST \), a two-tier system which works both at client and server side. On the server side, a knowledge base is updated on-line. On the client side, a plugin create a list of links to pages of interest. Our recommender system is shown to be privacy safe. No significant additional information, i.e. that could be used by malicious users, is needed to create a knowledge base. From this knowledge base, a set of web page clusters is extracted and used to build recommendations. More importantly, we show that the probability to guess whether a user has visited a set of pages on the basis of the extracted clusters, decrease exponentially with the cardinality of \(|U|\). This probability is the same both for any third party user and for the server providing this service as well. This means, that according to our framework, the server which collects information to build the knowledge base, can not breach users’ privacy. A set of experiments assess the quality of the recommendations. As the a future work, we want to tighten the bounds we provided in this article, and to study the evolution of the system from a privacy point of view. We had to evaluate the soundness of the system using historical data, but we are much more interested in how this scenario can change due to user interaction, i.e. when users actually use links provided by the system.

6. REFERENCES