Preserving Privacy in Web Recommender Systems

R. Baraglia\textsuperscript{1}, C. Lucchese\textsuperscript{1}, S. Orlando\textsuperscript{2,1}, R. Perego\textsuperscript{1}, F. Silvestri\textsuperscript{1}

\textsuperscript{1}HPC Lab, ISTI-CNR, Pisa, Italy
\textsuperscript{2}Dept. of Computer Science, Ca’ Foscari Univ., Venice, Italy

Abstract

The rapid growth of the Web has led to the development of new solutions in the Web recommender or personalization domain, aimed to assist users in satisfying their information needs.

The main goal of this chapter is to survey some of the recommender system proposals appeared in the literature, and to evaluate these proposals from the point of view of privacy preservation. Then, as an example of privacy-preserving approach for recommendations, we present $\pi$\textit{SUGGEST}, a privacy-enhanced system that allows for creating serendipity recommendations without breaching users privacy. $\pi$\textit{SUGGEST} helps users to navigate though a Web site, by providing dynamically generated links to relevant pages that have not yet been visited. The knowledge base on which the model used for making recommendations is built, is incrementally updated without tracking user sessions. This feature is particularly important when users do not trust the system, and do not want disclose their complete activity records or preferences. In this case, users may adopt techniques that avoid server-based session reconstruction, and that do not worsen the accuracy of the model extracted by $\pi$\textit{SUGGEST}. As an additional contribution, we show that $\pi$\textit{SUGGEST} does not allow malicious users to track or detect users activity or preferences.

1 Introduction

The goal of Web recommendation and personalization techniques is to “provide users with the information they want or need, without expecting from them to ask for it explicitly”\cite{19}.

Web Mining has shown to be a viable technique to discover information “hidden” into Web-related data \cite{11}. 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 \cite{14}. It can be used for different purposes such as recommendation, personalization, system improvement and site optimization.
A typical way to exploit WUM techniques for the recommendation task is the extraction of a model from usage data that permits to group users in different clusters, according to their interests, and to adaptively provide them useful recommendations. Obviously, the learning of a model from past usage data, or simply the collection of such data, can introduce privacy breaches, by either disclosing personal information or allowing malicious queries capable of reconstructing the knowledge collected by the system.

In this chapter, we first survey the techniques that have appeared in the literature and aim to automatically generate suggestions and recommendations to users, by also discussing their privacy preserving features.

In the second part of the chapter, we present our privacy-enhanced Web recommender system, called $\pi$SUGGEST, which is designed to dynamically generate personalized contents of potential interest for users of a Web Site, without providing privacy breaches to malicious users.

The architecture of $\pi$SUGGEST is based on a two-tier architecture. The server-based tier monitors and collects Web usage information in order to build a global model, whereas the client-based tier, to be plugged into the user browser, exploits this model to provide personalized recommendations to the single user. In particular, the plug-in is able to personalize on-the-fly a requested HTML page, by appending a list of page links (suggestions). The privacy-preserving strength of $\pi$SUGGEST relies on its capability of building an accurate model of Web usage, even if users hide and protect their own user sessions with the aim of not disclosing their complete activity records or preferences. The only information which is collected and maintained by the server are the single navigation hops of users, i.e., from browsing a given Web page to visiting the next one. The current user session is obviously maintained on the client-tier, which exploits it to give suggestions to the user, by comparing the current session to the usage model built and provided by the server.

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. In other words, we only discuss the case of malicious users who can make new HTTP requests to the server and analyze the supplied recommendations, and we do not deal with the case of malicious users who can access private information stored on the client-tier.

The quality of suggestions provided to users is evaluated by adopting the metric introduced in [3]. This metric tries to estimate the effectiveness of a recommendation system as the capacity of anticipating the requests that users will submit to the system farther in the future.

Summarizing, the main contributions of this chapter are:

- a comprehensive survey of the main recommender systems proposed in the literature, by also considering their possible privacy breaches;
- a new algorithm to incrementally generate users profiles in a privacy preserving way;
• a general privacy measure for Web recommender systems, which is used to evaluate how $\pi$SUGGEST is able to successfully preserve users privacy.

The rest of the chapter is organized as follows. In Section 2 we survey the main literature proposals concerning recommender systems, by also considering their possible privacy breaches. Section 3 presents the architecture and the algorithms used by $\pi$SUGGEST. Section 4 presents a framework for analyzing privacy in cluster-based recommender systems. This framework is then used for the analysis of $\pi$SUGGEST from the point of view of privacy preservation. Section 5 discusses the quantitative evaluation of the accuracy of the suggestions made by $\pi$SUGGEST. Finally, Section 6 concludes the chapter.

2 Taxonomy of Web Personalization and Recommendation

In general, a recommendation system accomplishes two tasks. The first is to collect data about users in order to understand the frequent patterns of interests. The second, is to provide interesting recommendations related to the current activity of the user.

Concerning the employed usage data, such data can be directly collected by logging the actions of users/customers who voluntarily registered in a collaborative service and accepted their preferences/interests/buying information to be recorded. This is the typical case of e-commerce and social networking websites.

Sometimes, the users are not willing to register, or simply they cannot. This is the most general case of Web-site recommender systems. However, similar data can be obtained from server logs. The quality of this data might be lower, mainly due to the uncertainty in distinguishing the identities of different users to build their profiles.

We do not address the problem of dealing with this type of raw data, but we need to introduce the useful concept of user session, which is the ordered set of actions performed by the user to accomplish a goal. Examples of sessions are the set of product description pages visited before actually buying an item, or the set of queries submitted to a Web search engine before landing on the desired page. Detecting user sessions is a complex task, since they are usually interleaved and hierarchically structured [10]. Nevertheless, several methodologies and techniques can be found in literature.

If, in addition to session it is possible to identify users, then sessions associated with a given user can be aggregated, thus yielding to a user profile.

The collection of user data, the management of user profiles and the generation of personalized recommendations raise a number of privacy issues. This is a quite wide topic, since privacy aspects are manifold, including several laws in different countries. Here we will focus on two aspects only, called privacy risks:

• **Risk I**: how are sensitive data collected and maintained?
• **Risk II**: is the recommendation process introducing any privacy breach?
Concerning the first aspect, it is clear that users must disclose several information about their activity, that is preferences/interests/buying information. This can be done with different level of detail, ranging from a very general profile, to fine-grained temporal monitoring, from anonymous interaction to a logging-in requirement.

Concerning the second aspect, that is the privacy risks of recommender systems, a preliminary paper [24] tried to formalize the problem and to measure the amount of privacy provided to users. In the paper, a recommender system is proposed that considers similar users that share a similar profile, e.g. if they share at least $w$ identical ratings. Given this similarity relationship, a social network over users data can be built by linking similar users. This network will be likely to be naturally clustered, so that it can allow to detect similar habits among groups of users. Once a user enters the system, recommendations based on the ratings previously expressed by other “similar” users can be provided for the items not been already visited (or purchased) by the user herself.

Dispensing recommendations in such a way turns out to constitute 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 $\{a, b, c, d\}$ and receives as a recommendation item $e$. Then, we know that there is a bunch of users who actually rated items in $\{a, b, c, d, e\}$. 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 training, 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.

In the following section, we survey the techniques proposed in literature to automatically generate suggestions and recommendations, and, in some cases, personalization of a user service. Differently from other surveys [1, 18], we will also take privacy issues into account. For each technique, we illustrate and discuss the following characteristics:

- **Model Building**: how information to understand user interests are gathered and stored.

- **Recommendation Generation**: how recommendations are generated as a response to user interaction.

- **Sensitive Data**: the personal data collected by the system.

- **Privacy Preservation**: the privacy concerns and guarantees.
To the best of our knowledge, the only recommender system taking into account both the two risk types discussed above is $\pi$SUGGEST, which is illustrated in detail in Section 3.

2.1 Content-Based Filtering

Model Building. This method [20, 13] works by collecting the content descriptions of all the items that a user has previously expressed interest for. By aggregating and weighting the set of features extracted from all such contents, we can build a user profile.

For example, in Web site recommendation we can consider the content of pages visited by a user, and collect all the terms appearing in these pages. These textual features can be opportunely weighted to build a weighted term vector representing the user profile. Analogously, in electronic commerce, items have an associated textual description that can be exploited in a similar way. Finally, in the context of query suggestion, it is possible to describe a query with the text of the pages returned by a search engine and clicked by users.

Recommendation Generation. To generate a recommendation, the user profile is compared with all the objects not already seen by the user. This comparison produces a ranked list of items, and consequently the highest ranked items may be suggested. To this aim, methods like cosine similarity can be exploited.

This approach is mainly used as a filtering process. Given a set of candidates, those non relevant to the user profile are pruned and the other are returned to the user, possibly with an associated rank. An example is [12], where a browser plugin highlights interesting links during the navigation.

A problem of this approach is the impossibility to provide serendipitous recommendations. Given the user profile, only very similar items are used to build suggestions, and therefore, such recommendations do not bring any new information to the user.

Sensitive Data. The system records every expression of interest, and therefore most of the activity of a user, e.g. items rated or pages visited, is monitored. However, if this method is used to implement a filtering process, such sensitive data can be kept on the client computer.

Privacy Preservation. If realized as a filtering based approach, it can be entirely implemented on the client side. The activity of the user is monitored in order to build the model, and a set of candidate items, resulting from the user activity, is filtered and possibly ranked. Since no information flows outside the user client, no privacy concern is raised. However, the applicability is limited since only passive filtering is applied.

An alternative approach [30] stores the user profile on a centralized server. This is applied to a web search engine, where the profile is used to re-rank results.
of submitted queries. To achieve some privacy guarantee, the user is allowed to exclude from the profile the least frequent terms. The underlying assumption is that infrequent terms may identify a user, while frequent ones let him hide in the crowd. There is a trade-off, since the least frequent terms are obviously important in the recommendation process.

### 2.2 Collaborative Filtering

#### Model Building.
The model consists of a global $\text{Users} \times \text{Items}$ matrix, where each element $(i, j)$ stores the rating of item $i$ given by user $j$ [9]. Such matrix describes all the preferences expressed by all the users.

Depending on the application, the rating can be explicit or implicit. In electronic commerce, the items correspond to purchased products along with their discrete ratings (e.g. one to five stars) assessing product quality or user satisfaction. In a Web scenario, items are the Web pages visited by a user. The implicit item rating may be measured by considering the number of page visits, or the cumulative time spent on the page.

One of main issues is related with the data sparsity. Since the number of user ratings is usually small with respect to the number of possible items, the model will contain mostly missing values, which makes it difficult to generate relevant recommendations [27].

#### Recommendation Generation.
The profile of a user is defined as the set of ratings for the visited/purchased/seen items. By using a $k\text{NN}$ search, it is possible to identify the $k$ most similar user profiles in the model. Then, the most popular items in such profiles are used to build recommendations.

Scalability and churn are two important issues [7]. The scalability is related to the exploitation of a lazy $k\text{NN}$ classifier on a large $\text{Users} \times \text{Items}$ matrix. The churn is related to the possibly large amount of items that can appear over time, e.g. in a news personalization system. Also, new items and new users are associated with a very small number of ratings, and therefore they can hardly play a role in the recommendation process.

Clustering and, more generally, model-based algorithms can be used to improve the efficiency and scalability of the real-time personalization tasks. By grouping together similar users and similar terms, it is possible to reduce the complexity of the recommendation process.

#### Sensitive Data.
As for content-based filtering, the activity of a user is monitored. In this case, however, the recommendation is based on a search operation for similar user profiles, and therefore such information is stored, maintained and indexed in a centralized server.

#### Privacy Preservation.
The method suffers from privacy breaches, since users have to grant to third parties personal information about their preferences, in order to speedup $k\text{NN}$ search operation.
There exists a proposal [6] that uses a blend of secure multiparty computation and factor analysis. The model consists in a matrix $\text{Users} \times \text{Topics}$, where the topics and the mapping between items and topics are automatically learned through factor analysis. However, user profiles are not shared to build the model. Rather, a secure multiparty computation technique is used to learn the model by exchanging encrypted messages. At the end of the process, every user has a copy of the model. A drawback of the method is that several customers must be on-line in order to participate simultaneously in the computation and subsequent model construction. While the approach is safe for what regards risk I, it does not deal with risk II.

There are some other proposals that try to reduce somewhat risk I. For example, by avoiding collecting all data in a single site, or by perturbing sensitive data regarding user activities. In [2] the authors propose to split customer data between the merchant and a semi-trusted third party. For example, this other party could be trusted to only maintain the customers’ demographic information, while the merchant to only manage item ratings. A proposal that still aims to split users data among many parties appeared in [23]. In that paper these parties hold disjoint sets of items ratings collected by the same user, but the devised privacy-preserving protocol is able to supply a recommendation service using their joint data without disclosing them to each other.

An interesting proposal [22] provide the perturbation of users ratings before submitting them to a central database on which the collaborative filtering algorithm is ran. Of course the amount of privacy is directly proportional to the amount of perturbation, whose drawback is a reduction of the accuracy of recommendations achieved.

### 2.3 Item-based Collaborative Filtering

**Model Building.** The method adopts the same user-based profiles of standard collaborative filtering [26], thus building the same $\text{Users} \times \text{Items}$ matrix.

**Recommendation Generation.** Rather than finding similarities among users, the objective is to find similarities among items. Two items are considered similar if they share similar ratings, that is if the corresponding columns of the $\text{Users} \times \text{Items}$ matrix are similar. Differently from content-based collaborative filtering, the content/description of objects does not affect the similarity.

In order to make a suggestion, a target item is needed, e.g. the product the user has just purchased. The method suggests items not previously seen that are in the neighborhood of the target item.

**Sensitive Data.** For each visited/purchased/seen item, we need to collect its ratings by all the users, like in a classical collaborative filtering method.

**Privacy Preservation.** The basic units of information, on the basis of which the method makes its recommendations, exactly corresponds to the ratings of an
item by all the users. The need of collecting, for each item, the private ratings of all the users, constitutes a clear privacy breach.

2.4 Recommending by Clustering Unordered User Sessions

Model Building. In the previous approaches we have seen different models that do not account for the temporal aspect of each single user interaction. The sequence of all the ratings of each user is flattened in a single set, as if they occurred simultaneously.

It is possible to see the interactions of a user with the system as a set of sessions, where a session is an ordered sequence of items visited by a user within a given time interval. A session thus contains items that can be considered as related from the user-side.

Many techniques take into account sessions, but do not exploit the sequential ordering of items within each session [31, 15]. A session is simply considered an n-dimensional vector, where the i-th element is the weight or degree of interest.

In [16], the Association Rule Hypergraph Partitioning (ARHP) technique is proposed. The idea is to group together items that frequently occur together in users session. The frequent itemsets, i.e. sets of items that occur not less than a given threshold in user sessions, returned by an Apriori-like algorithm are considered as hyperedges of a hypergraph. A hypergraph is an extension of a graph in the sense that each hyperedge can connect more than two vertices. A set of clusters is finally obtained by recursively partitioning the hypergraph into components with high connectivity. Finally, a cluster of items can be characterized by its median, or any other aggregation of the profiles of the various items in the cluster.

The whole model building process can be performed off-line, before and independently of the on-line recommendation generation. Whereas this allows more expensive model generations to be exploited, aiming to provide more accurate suggestions, we have to deal with the aging of the model built.

Recommendation Generation. The active user session is used to generate recommendations. As the user continues its activity, the user profile is updated by aggregating information about the items being viewed. This profile is compared with the cluster representatives in the model. The items in the most similar cluster, which were not already visited, are used to create recommendations [31].

Depending on the application, different similarity measures can be exploited. In most cases, a good measure is simply counting the number of items the user has accessed in each cluster: a cluster matches the profile if the count is above a certain threshold.

Sensitive Data. We need to collect user sessions, that is the groups of items visited by each user during a given time interval. In particular, the complete
sessions to build the model, and the partial ones to make a recommendation.

Privacy Preservation. In this approach, neither risk I nor risk II are considered. The user is monitored during her activity, and the data contained in her sessions managed by the centralized recommender system.

2.5 Recommending through Association Analysis of Unordered User Sessions/Profiles

Model Building. Association rules are a powerful tool for the discovery of strong dependencies between sets of items in a database. A model consists of a set of rules, with given support (number of times the rule occurs in the database) and confidence (conditional probability of the consequent given the antecedent).

In the context of recommender systems, it would be possible to mine for association rules by considering sessions as transactions, thus discovering correlation between sets of items, pages, queries, etc. Such rules, in conjunction with the activity or profile of a target user, can be used for recommendation generation.

Also in this case, the model construction is performed off-line.

Recommendation Generation. We illustrate two scenarios: collaborative filtering and query recommendation. In the first scenario, the preferences of the target user are matched against the items in the antecedent of each rule, and the items in the consequent are used to create recommendations. The confidence of rules is used to rank recommendations [25].

In the second scenario, only rules with a single query both in the antecedent and in the consequent are considered. The query submitted by the user is searched among rules antecedents. The matching rules, and thus the queries in the rules consequent are sorted according to confidence, and used to build the recommendations [8].

Sensitive Data. Also in this case, the data contained in the user session must be sent to a centralized server, where they are used to build the model. The same remarks made above and concerning the exploitation of unordered user sessions hold.

Privacy Preservation. The same remarks made above concerning the exploitation of unordered user sessions for building a model and recommending hold in this case.

2.6 Recommending by Clustering Ordered User Sessions

Model Building. In addition to considering users sessions, it is possible to take into account also the order of the items in each session. We illustrate two
algorithms for web page recommendation, SUGGEST [3, 5] and PageGather [21], that find clusters of pages based on this idea.

The model built by the two algorithms is an undirected graph whose vertices are the pages visited by the users so far, and the edges represent the jump through a link from one page to another. Note that the graph is undirected, meaning that a link is interpreted as a symmetric similarity between two pages.

PageGather needs the whole collection of the user sessions to build the graph. The arc \((i,j)\), is labelled with \(\min P(i|j), P(j|i)\), where \(P(j|i)\) is the conditional probability that page \(j\) is visited during a session given that page \(i\) has been visited in the same session.

SUGGEST uses less information than PageGather from the various sessions to build the graph. Each undirected arc \((i,j)\) is labeled with the number of times pages \(i\) and \(j\) have been accessed consecutively, in any order, by a user. Analogously, graph nodes are labeled with the number of times the associated page has been visited. Note that such graph labeling can be carried out even if the complete user session is not available. It is sufficient to know the referral of each requested page.

Both algorithms remove noise by dropping arcs with small weight. Finally, graph clustering algorithms are used, aimed at finding the connected components of the graph, i.e. corresponding to clusters of pages frequently occurring in users visiting paths.

Recommendation Generation. SUGGEST provides an on-line component that keeps track of the last recently visited pages, and suggests new pages belonging to the cluster with the largest intersection with these pages.

In PageGather there is no on-line component. It simply generates from the clusters a static index of correlated pages which are kept in a separate “Suggestion Section” of the web-site.

Sensitive Data. PageGather and SUGGEST exploit the information regarding the page visiting order in the user sessions to build the graph used to cluster the pages. Indeed, SUGGEST only needs to collect single page transitions from each session. A user session composed of \(n\) page visits is thus completely equivalent to \(n-1\) short sessions made by distinct users, and visiting the same pages.

Privacy Preservation. The method exploited by PageGather suffers from privacy breaches, since complete (ordered) user sessions need to be collected, although the identities of users can be preserved, so that global user profiles cannot be reconstructed by joining different sessions. On the other hand, SUGGEST only need to collect user transitions from one page to another.
2.7 Recommending through Sequential Analysis of Ordered User Sessions/Profiles

Model Building. So far we have discussed the use of frequent itemsets, and the resulting association rules. These are the least restrictive type of navigational patterns, since they take into account the presence of items in session, disregarding their order.

When considering a session as an ordered set of items, there are a number of data mining tools able to extract significant patterns that take into account that order. In particular, it is possible to use sequential patterns and contiguous sequential patterns to analyze users navigational trails [29, 17]. A sequential patterns $S$ is an ordered set of items, and it requires all of its items to appear in the same temporal order. In addition, a contiguous sequential pattern requires its items to be adjacent in the users sessions.

The resulting set of patterns can be used as a model of frequent users behavior. In order to perform efficient operation on the model, it can be stored with a trie data structure.

Recommendation Generation. As usual, recommendations stem from the current session of a given user. Given the ordered set of items recently visited (purchased, etc.), this can be used to traverse the trie of the frequent sequential patterns. If the current profile matches part of a profile stored in the model, then the remaining items are used to create recommendations.

In [17] contiguous sequential patterns were judged as too much restrictive in the general context of recommendation generation, even if very valuable in page prefetching applications.

Sensitive Data. Similarly to the other session-aware algorithms, the activity of the user must be continuously monitored in order to build/update the model and to generate recommendations.

Privacy Preservation. The same remarks made above concerning the exploitation of ordered user sessions for building a model and recommending hold in this case.

3 The $\pi$SUGGEST system

$\pi$SUGGEST, is an evolution of SUGGEST [3, 5], the on-line recommender mentioned in Section 2.6. The main novelty of $\pi$SUGGEST is that its two components, the one that updates the knowledge base, and the one that builds and provides recommendations, are well separated (see Figure 1). The former is placed on the web server (it is indeed a module of the Apache Web server). The latter runs on the client-side as a browser plug-in.

In order to collect information about navigational patterns, $\pi$SUGGEST does not need to maintain the complete user sessions. It only needs to manage
an undirected graph $G = (V,E)$ with weighted edges. Each vertex $v_i = V$ corresponds to a page hosted by the Web site. Since in our model the interest in a page depends on its visiting order during the various sessions, each edge $e_{ij} = E$, which connects vertices $v_i$ and $v_j$, is associated with a weight $W_{ij} = N_{ij}/\max\{N_i, N_j\}$. $N_{ij}$ is the number of times the two pages, corresponding to $v_i$ and $v_j$, have been accessed consecutively (and in any order) by the user community, while $N_i$ and $N_j$ are, respectively, the number of times the same two pages have been visited. We divide by $\max\{N_i, N_j\}$ since we want to reduce the relative importance of links involving index pages. Generally, even if such pages do not contain useful content, they are used as a starting point for a browsing session. Moreover, users often return back to these pages several times, in order to start the visit of a new branch of the Web site. Therefore, though it is very likely that index pages are visited along with any other page, nevertheless they are of little interest as potential suggestions.

A triangular adjacency matrix $N$ is indeed used to store the knowledge base corresponding to graph $G$: each entry $N[i,j]$, $i \neq j$, stores $N_{ij}$, while each entry $N[i,i]$ stores $N_i$. The adjacency matrix is incrementally maintained by the $\pi$SUGGEST component on the server-side (see Figure 1), by only considering single HTTP requests coming from clients. Note that each HTTP request contains both the URL of the requested page and the referral one, i.e. the page from which the user is coming. The server-side component of $\pi$SUGGEST exploits the adjacency matrix to find disjoint clusters of strongly related pages. In particular, it partitions $G$ on the basis of its connected components, by using a modified version [3, 5] of the well known incremental connected components
algorithm [28]. The algorithm is driven by two threshold parameters, aiming to limit the number of edges to visit, but also to avoid the generation of clusters that may be statistically irrelevant (because they might over-fit the knowledge base). In particular,

1. we filter out from $G$ all the edges whose weight $W_{i,j}$ is below the constant $\text{minfreq}$. The pairs of pages connected by such edges are indeed poorly correlated, and thus are not considered by our clustering algorithm;

2. we only take into consideration those connected components whose size is greater than a fixed number of nodes, namely $\text{minclustersize}$. All the other components are indeed discarded because considered not significant enough.

The extracted information, i.e., the cluster identifiers together with the various vertices/pages of $G$, is maintained in another a vector $L$. Since in large Web sites the size of matrix $N$ and vector $L$ might exceed the maximum available main memory, the server-side component of $\pi_{\text{SUGGEST}}$ adopts an LRU-based strategy to store in main memory only the portions of the data structures associated with those pages that have been recently accessed by users.

As illustrated by Figure 1, the client-side component of $\pi_{\text{SUGGEST}}$ asks the server for the page clusters (stored in $L$) when a session starts. The same component is responsible for tracking the user and maintaining her/his session. It also builds suggestions by finding the cluster that has the largest intersection with the $\text{PageWindow}$ (i.e., the last portion of the current session). The suggestions only include the most relevant pages in the cluster, according to an order determined during the clustering phase.

**Privacy-preserving features of $\pi_{\text{SUGGEST}}$.** We have previously introduced two kinds of privacy risks. The first risk is due to the data collected by the recommender service, and the second risk is related with the recommendation generation process. The two-tier architecture of $\pi_{\text{SUGGEST}}$ makes it possible to overcome both these two kinds of risks.

The information collected by $\pi_{\text{SUGGEST}}$ is just a couple of web-page URLs for each user interaction: requested and referral pages. This allows the user to protect his privacy against the recommender system by adopting a number of technological solutions. Such techniques range from cookies-related stuff to methods aimed at masking or scrambling the client IP address\(^1\). In such a way, the user may achieve two goals: changing his identity at every request, and avoid the recommender system to reconstruct the user session.

Regarding the second risk type, the system needs the current user session in order to provide recommendations. Thanks to the two-tier structure, the user session is only built and kept locally at the client-side, and never communicated to the recommender service. Therefore, the server component of $\pi_{\text{SUGGEST}}$

\(^1\)See, for example, http://www.torproject.org/index.html.en.
is not aware of the user sensitive data, i.e. identity and sessions, but still it can build a model for the generation of good recommendations.

The model built by the centralized component is sent to every client, which uses the model to generate recommendations. The model is general enough to prevent any malicious user from deducing any sensitive information.

In the theoretical framework of $\pi$SUGGEST we are not considering those form of threats that may affect a specific user, his software or his hardware. For example, these issues may arise in context where the attacker is able to sniff the Internet traffic of a single client, or when the client browser is compromised so that an attacker can access to the complete user sessions. Finally, we are not considering the case where a significant number of clients performs artificial HTTP requests to sabotage the model building phase.

4 $\pi$SUGGEST and Privacy

In order to evaluate the privacy features of $\pi$SUGGEST, we start quantifying the level of confidence associated with the capability of inferring information about users activities.

In general, a recommender system tries to classify a user on the basis of the visited pages. Each class of users is associated with a subset of pages, which are of interest for them. In $\pi$SUGGEST, these subsets of pages (i.e., the clusters of pages) are a public information, since $L$ is returned to each client when a user session starts. In other systems, such class representatives are kept private, even if part of them are published in the form of user recommendations. We are interested in investigating which kind of information is revealed when information about the composition of a generic cluster is disclosed.

From the point of view of the plug-in on the client-side, a cluster is simply a set of pages $C = \{p_0, p_1, ..., p_q\}$, obtained by partitioning graph $G$ on the server-side. Cluster $C$ actually corresponds to a (partially or completely) connected component of $G$. However, the plug-in cannot be aware of which pairs of pages actually correspond to edges in the $G$ graph. On the other hand, a user activity (or session) corresponds to a set of visited pages. Since the user moved from a page to another, there must exist a partially (or completely) connected graph behind such set of pages.

Since we are interested in analyzing which kind of user activities may have generated a given cluster, it is useful to introduce the concept of valid cluster generator.

**Definition 1** Let $C = \{p_0, p_1, ..., p_q\}$ be a cluster of pages, and $U = \langle U_1, ..., U_n \rangle$ be a set of user activities. Each $U_i$ is a subset of pages belonging to $C$ that have been visited by some user. $U$ is a valid cluster generator iff the following three conditions hold:

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

Therefore, a cluster generator is the minimal set of user activities (sessions), that are able to create the connected component \( C \). We introduce minimality to avoid anomalous combinations that may be useless in this context. For example, we do not want the two sessions \( \langle \{abcd\}, \{abc\} \rangle \) to be a valid generator for cluster \( \{abcd\} \), since the cluster is also supported by the first session only.

**Definition 2** Given a cluster \( C = \{p_0, p_1, ..., p_q\} \), a valid cluster generator \( U \), and a recommender system \( \Sigma \), the privacy level \( \Pi \) of \( \Sigma \) with respect to \( U \) is:

\[
\Pi_{\Sigma}(U, C) = 1 - P(U \mid C)
\]

If we can estimate \( U \) with high probability on the basis of the knowledge of a cluster \( C \), the system has a very low level of privacy. On the other hand, if there is no \( U \) which is likely to be a generator of \( C \), then the system has a high level of privacy.

For example, suppose that the client-side plug-in of \( \piSUGGEST \) receives a cluster of pages, namely \( C = \{a, b, c, d, e\} \). Many different events may have generated \( C \). For example, a single user who visited all the pages \( \{a, b, c, d, e\} \), or two users who visited respectively the pages \( \{a, b, c\} \) and \( \{c, d, e\} \), or three users who visited the pages \( \{a, b, c\}, \{a, c, d\} \) and \( \{d, e\} \), and so on. Note that different users activities may have generated not only the same cluster, but also the same knowledge base.

Although this example is very small, we were able to find a lot of valid cluster generators. Before considering this example more formally, let us consider clusters of smaller sizes. If \( |C| = 2 \), since \( \piSUGGEST \) creates an edge between two pages if and only if they were visited consecutively, we can conclude that some user visited the two pages with probability 1, and therefore our privacy level is \( 1 - 1 = 0 \). Clearly, we only have one acceptable user activity and thus no privacy. For the case \( |C| = 3 \), we have only four valid cluster generators (the three subsets of two elements of \( C \), and the set \( C \) itself) leading to a privacy level of \( 1 - 1/4 = 0.75 \). However, the recommendations provided by using these “small” clusters are of little significance. Moreover, they would lead to the generation of an over-fitted model with respect to the training data. For this reason, our system builds clusters whose cardinality is greater than or equal to 4.

**Theorem 1** Given a cluster \( C = \{p_1, ..., p_q\} \), where \( q \geq 4 \), and a valid cluster generator \( U \), the privacy level \( \Pi \) provided by \( \piSUGGEST \) can be bounded, and its lower bound is:

\[
\Pi_{\piSUGGEST}(U, C) = 1 - P(U \mid C) \geq 1 - \frac{1}{2^{|C|}}
\]
The previous theorem, whose proof can be found in [4], states that the amount of possible valid cluster generators is very high. Therefore this makes it impossible 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 user activities as above, but also a confidence level for the activity of a single user.

**Definition 3** Given a cluster \( C = \{p_0, p_1, \ldots, p_n\} \), let \( U = \{q_0, q_1, \ldots, q_n\} \), \( U \subseteq C \), be the set of pages visited by a single user. The privacy level \( \Pi^* \) provided by a recommender system \( \Sigma \) with respect to \( U \) is:

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

Given that the system created and suggested cluster \( C \), we want to weigh the chance that some users have actually visited a set of pages \( U \), where \( U \subseteq C \).

**Theorem 2** Given a cluster \( C = \{p_0, p_1, \ldots, p_q\} \), and a set of pages \( U = \{q_1, \ldots, q_h\} \) visited by a user, where \( U \subseteq C \), the privacy level \( \Pi^* \) provided by \( \pi_{SUGGEST} \) with respect to \( U \) can be lower bounded, and the bound is:

\[
\Pi^*_{\pi_{SUGGEST}}(U, C) = 1 - P(U \mid C) \geq 1 - \frac{1}{3^{\frac{|C|}{2}}}
\]

Interested readers can refer to [4] for the proof of the previous theorem.

Theorem 1 and Theorem 2 state that if the \( \pi_{SUGGEST} \) system is plugged into a privacy safe system, it will not provide any privacy breach. We say that a system is **privacy safe** if two conditions hold: (i) the user activity cannot be tracked, (ii) the user activity cannot be inferred. Condition (i) holds by definition in a safe system. Moreover, neither publishing the clustered structure can be considered a privacy breach, even if it could be inferred with consecutive queries to the system. Theorem 1 assures that the privacy provided by \( \pi_{SUGGEST} \) 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.

**Discussion.** The classification-based approach, which is used by many popular recommender systems, could be a privacy breach by itself. It may disclose to a malicious person which pages a group of users have actually visited.

In \( \pi_{SUGGEST} \) we have defined a new privacy measure that 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 disclosed (under the form of recommendation) by the system. Finally, we have introduced a two-tier system for privacy-enhanced recommendation. On the server-side, a knowledge base is updated on-line. On the client-side, a plug-in creates a list of links to pages of interest.
\(\pi\text{SUGGEST}\) has been shown to be privacy safe. From its knowledge base, a cluster \(C\) of web pages is extracted and used to build recommendations. The probability to guess whether a user has visited a set of pages \(U, U \subseteq C\), on the basis of the extracted cluster only, 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. In other words, also the server that collects information to build the knowledge base cannot breach users’ privacy.

5 \(\pi\text{SUGGEST}\) Evaluation

Measuring the quality of a recommendation systems is considered a very difficult task. We have to characterize the quality of the suggestions obtained, by quantifying how useful the suggestions are for the users.

The \(\pi\text{SUGGEST}\) effectiveness can be evaluated by using the performance measure introduced in [3], which is based on the intersection of real user sessions with the corresponding set of suggestions. For every session \(S_i\), and a set of suggestions \(R_i\) provided by the system, we could derive the quality of suggestion by using:

\[
\omega_i = \frac{|S_i \cap R_i|}{|S_i|}
\]

Unfortunately, this simple measure cannot capture the potential impact of the suggestions on the user navigational session. For example, suppose that a page that the user would visit at the end of the session is instead suggested at the beginning of the session: in this case the suggestion should be very valuable for the user, who can find a shorter way to what s/he is looking for. Therefore we can extend expression 1 by taking into account the distance between the suggestions and the actual pages visited during the session. To this end, we need to split \(S_i\) into two halves. Only the first half \(S^1_i\) is used to generate the set of suggestions \(R_i\). The second half is instead used to measure the intersection with the suggestions. For every page \(p_k \in S^2_i \cap R_i\), where \(p_k\) appears in position \(k\) within \(S^2_i\), we add a weight \(f(k)\). We choose \(f\) so that more importance is given to pages actually visited at the end of the session. In conclusion, for the whole session log, we can measure the quality of the suggestions by:

\[
\Omega = \frac{1}{F} \sum_{i=1}^{N_S} \sum_{k=1}^{|S^2_i|} \left[ p_k \in (S^2_i \cap R_i) \right] f(k)
\]

where \(N_S\) is the number of sessions, \([expr]\) is the truth function (which is equal to 1 if \(expr\) evaluates to True, 0 otherwise), while \(F\) is a normalization factor on the weights.

For a quantitative evaluation of \(\pi\text{SUGGEST}\), refer to [3], where we used three real-life access logs\(^2\): Berkeley, NASA, USASK, produced by the Web servers of

\(^2\)www.web-caching.com
the Computer Science Department of Berkeley University, Saskatchewan University and Kennedy Space Center, respectively. In these experiments, we chose \( f(k) = k \), so that the page weights increase linearly with the corresponding positions into the session. For each test, we generated requests to an Apache server running \( \pi SUGGEST \), and recorded the suggestions generated for every navigation session contained within the access log file considered.

For each log file, we measured \( \Omega \) as a function of the \( \minfreq \) parameter, which is used to filter out from the graph \( G \) all the “infrequent edges”. Suggestions generated by \( \pi SUGGEST \) show a higher quality than a random generator of suggestions, and this quality reaches the maximum for \( \minfreq = 0.2 \) for almost all the log files.

6 Conclusions

In this chapter we have described the distinguishing features of privacy preserving recommender systems, and have discussed the main features and evaluation methodologies of a privacy-preserving Web recommender system.

The chapter is divided into two main parts. In the first part we introduce the problem, survey the approaches existing in literature, and highlight the implications from a privacy point of view. For each technique we illustrate how models are built, how recommendations are generated, what data are retained and stored, and, finally, privacy concerns and guarantees it offers. In particular, we point out two different kinds of risks for privacy that are related to what kind of data is retained, and what kind of breach for privacy the recommendation models are subjected to.

Regarding the two risks just mentioned, in the second part of the chapter we present \( \pi SUGGEST \), a recommender system that has been specifically designed to address and overcome those risks. \( \pi SUGGEST \) builds upon a previous work on online web recommender systems \([3, 5]\) and addresses the problem of privacy preservation using a two-tier architecture. The major difference with previous versions of the system is that information about users’ sessions and visited pages is stored only on the client-side. This limits considerably the amount of knowledge that can be inferred by querying the knowledge base (stored at server-side), and this makes practically impossible to disclose navigational information about single users.

There are some open questions that will become important in the next future as recommender systems will be more and more used. It is very important to think of how to construct data repositories able to manage large numbers of subjects and objects along with actions performed by those subjects on the objects stored. As an example consider the effort to cope with the large amount of new object ratings. Privacy concerns, in this cases, are very important. Furthermore, related topics like: advertising, reputation based discovery, and other processes will have to be able to model and identify, in a privacy preserving manner, dynamic trends, such as emerging “topics”, and to take these dynamic behaviors over time into account in making recommendations. Privacy (as well
as trust enforcement) mechanisms are an important requirement for making recommender system usable. Users not feeling protected enough from privacy breaches, will stop them.

References


