Supporting Privacy Protection in Personalized Web Search

Lidan Shou, He Bai, Ke Chen, and Gang Chen

Abstract—Personalized web search (PWS) has demonstrated its effectiveness in improving the quality of various search services on the Internet. However, evidences show that users' reluctance to disclose their private information during search has become a major barrier for the wide proliferation of PWS. We study privacy protection in PWS applications that model user preferences as hierarchical user profiles. We propose a PWS framework called UPS that can adaptively generalize profiles by queries while respecting user-specified privacy requirements. Our runtime generalization aims at striking a balance between two predictive metrics that evaluate the utility of personalization and the privacy risk of exposing the generalized profile. We present two greedy algorithms, namely GreedyDP and GreedyIL, for runtime generalization. We also provide an online prediction mechanism for deciding whether personalizing a query is beneficial. Extensive experiments demonstrate the effectiveness of our framework. The experimental results also reveal that GreedyIL significantly outperforms GreedyDP in terms of efficiency.

Index Terms—Privacy protection, personalized web search, utility, risk, profile

1 Introduction

The web search engine has long become the most important portal for ordinary people looking for useful information on the web. However, users might experience failure when search engines return irrelevant results that do not meet their real intentions. Such irrelevance is largely due to the enormous variety of users' contexts and backgrounds, as well as the ambiguity of texts. *Personalized web search* (PWS) is a general category of search techniques aiming at providing better search results, which are tailored for individual user needs. As the expense, user information has to be collected and analyzed to figure out the user intention behind the issued query.

The solutions to PWS can generally be categorized into two types, namely *click-log-based* methods and *profile-based* ones. The click-log based methods are straightforward—they simply impose bias to clicked pages in the user's query history. Although this strategy has been demonstrated to perform consistently and considerably well [1], it can only work on repeated queries from the same user, which is a strong limitation confining its applicability. In contrast, *profile-based* methods improve the search experience with complicated user-interest models generated from user profiling techniques. Profile-based methods can be potentially effective for almost all sorts of queries, but are reported to be unstable under some circumstances [1].

Although there are pros and cons for both types of PWS techniques, the *profile-based* PWS has demonstrated more effectiveness in improving the quality of web search recently, with increasing usage of personal and behavior information to profile its users, which is usually gathered implicitly from query history [2], [3], [4], browsing history [5], [6], click-through data [7], [8], [1] bookmarks [9], user documents [2], [10], and so forth. Unfortunately, such implicitly collected personal data can easily reveal a gamut of user's private life. Privacy issues rising from the lack of protection for such data, for instance the AOL query logs scandal [11], not only raise panic among individual users, but also dampen the data-publisher's enthusiasm in offering personalized service. In fact, privacy concerns have become the major barrier for wide proliferation of PWS services.

1.1 Motivations

To protect user privacy in profile-based PWS, researchers have to consider two contradicting effects during the search process. On the one hand, they attempt to improve the search quality with the personalization utility of the user profile. On the other hand, they need to hide the privacy contents existing in the user profile to place the privacy risk under control. A few previous studies [10], [12] suggest that people are willing to compromise privacy if the personalization by supplying user profile to the search engine yields better search quality. In an ideal case, significant gain can be obtained by personalization at the expense of only a small (and less-sensitive) portion of the user profile, namely a *generalized* profile. Thus, user privacy can be protected without compromising the personalized search quality. In general, there is a tradeoff between the search quality and the level of privacy protection achieved from generalization.

Unfortunately, the previous works of privacy preserving PWS are far from optimal. The problems with the existing methods are explained in the following observations:

1. *The existing profile-based PWS do not support runtime profiling.* A user profile is typically generalized for

• *The authors are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, P.R. China. E-mail: should@acm.org, bailaohe@gmail.com, {chenk, cg}@zju.edu.cn.*

Manuscript received 1 May 2012; revised 29 Aug. 2012; accepted 3 Oct. 2012; published online 11 Oct. 2012.
Recommended for acceptance by B.C. Ooi.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2012-05-0302.
Digital Object Identifier no. 10.1109/TKDE.2012.201.

only once offline, and used to personalize all queries from a same user indiscriminatingly. Such "one profile fits all" strategy certainly has drawbacks given the variety of queries. One evidence reported in [1] is that profile-based personalization may not even help to improve the search quality for some ad hoc queries, though exposing user profile to a server has put the user's privacy at risk. A better approach is to make an online decision on

a. whether to personalize the query (by exposing the profile) and
b. what to expose in the user profile at runtime.

To the best of our knowledge, no previous work has supported such feature.

2. *The existing methods do not take into account the customization of privacy requirements.* This probably makes some user privacy to be overprotected while others insufficiently protected. For example, in [10], all the sensitive topics are detected using an absolute metric called *surprisal* based on the information theory, assuming that the interests with less user document support are more sensitive. However, this assumption can be doubted with a simple counterexample: If a user has a large number of documents about "sex," the *surprisal* of this topic may lead to a conclusion that "sex" is very general and not sensitive, despite the truth which is opposite. Unfortunately, few prior work can effectively address individual privacy needs during the generalization.
3. *Many personalization techniques require iterative user interactions when creating personalized search results.* They usually refine the search results with some metrics which require multiple user interactions, such as *rank scoring* [13], *average rank* [8], and so on. This paradigm is, however, infeasible for runtime profiling, as it will not only pose too much risk of privacy breach, but also demand prohibitive processing time for profiling. Thus, we need predictive metrics to measure the search quality and breach risk after personalization, without incurring iterative user interaction.

1.2 Contributions

The above problems are addressed in our UPS (literally for User customizable Privacy-preserving Search) framework. The framework assumes that the queries do not contain any sensitive information, and aims at protecting the privacy in individual user profiles while retaining their usefulness for PWS.

As illustrated in Fig. 1, UPS consists of a nontrusty search engine server and a number of clients. Each client (user) accessing the search service trusts no one but himself/herself. The key component for privacy protection is an *online profiler* implemented as a search proxy running on the client machine itself. The proxy maintains both the complete *user profile*, in a hierarchy of nodes with semantics, and the *user-specified (customized) privacy requirements* represented as a set of *sensitive-nodes*.

The framework works in two phases, namely the *offline* and *online* phase, for each user. During the offline phase, a hierarchical user profile is constructed and customized with the user-specified privacy requirements. The online phase handles queries as follows:

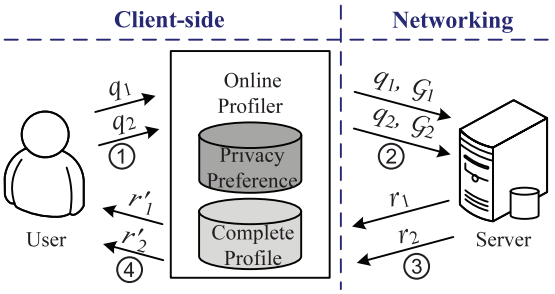


Fig. 1. System architecture of UPS.

1. When a user issues a query q_i on the client, the proxy generates a user profile in runtime in the light of *query terms*. The output of this step is a *generalized* user profile $\mathcal{G}_i$ satisfying the privacy requirements. The generalization process is guided by considering two conflicting metrics, namely the *personalization utility* and the *privacy risk*, both defined for user profiles.
2. Subsequently, the query and the generalized user profile are sent together to the PWS server for personalized search.
3. The search results are personalized with the profile and delivered back to the query proxy.
4. Finally, the proxy either presents the raw results to the user, or reranks them with the complete user profile.

UPS is distinguished from conventional PWS in that it 1) provides runtime profiling, which in effect optimizes the personalization utility while respecting user's privacy requirements; 2) allows for customization of privacy needs; and 3) does not require iterative user interaction. Our main contributions are summarized as following:

- We propose a privacy-preserving personalized web search framework UPS, which can generalize profiles for each query according to user-specified privacy requirements.
- Relying on the definition of two conflicting metrics, namely *personalization utility* and *privacy risk*, for hierarchical user profile, we formulate the problem of privacy-preserving personalized search as δ-Risk Profile Generalization, with its $\mathcal{NP}$-hardness proved.
- We develop two simple but effective generalization algorithms, GreedyDP and GreedyIL, to support runtime profiling. While the former tries to maximize the *discriminating power (DP)*, the latter attempts to minimize the *information loss (IL)*. By exploiting a number of heuristics, GreedyIL outperforms GreedyDP significantly.
- We provide an inexpensive mechanism for the client to decide whether to personalize a query in UPS. This decision can be made before each runtime profiling to enhance the stability of the search results while avoid the unnecessary exposure of the profile.
- Our extensive experiments demonstrate the efficiency and effectiveness of our UPS framework.

The rest of this paper is organized as follows: Section 2 reviews the related work, focusing on PWS and its privacy preservation. Section 3 introduces some preliminary knowledge and gives the problem statement. Section 4 presents the procedures of UPS framework. The generalization techniques used in UPS are proposed in Section 5. Section 6 further discusses some implementation issues of UPS. The experimental results and findings are reported in Section 7. Finally, Section 8 concludes the paper.

2 Related Works

In this section, we overview the related works. We focus on the literature of profile-based personalization and privacy protection in PWS system.

2.1 Profile-Based Personalization

Previous works on profile-based PWS mainly focus on improving the search utility. The basic idea of these works is to tailor the search results by referring to, often implicitly, a user profile that reveals an individual information goal. In the remainder of this section, we review the previous solutions to PWS on two aspects, namely the *representation* of profiles, and the *measure* of the effectiveness of personalization.

Many profile representations are available in the literature to facilitate different personalization strategies. Earlier techniques utilize term lists/vectors [5] or bag of words [2] to represent their profile. However, most recent works build profiles in hierarchical structures due to their stronger descriptive ability, better scalability, and higher access efficiency. The majority of the hierarchical representations are constructed with existing weighted topic hierarchy/graph, such as ODP[1] [1], [14], [3], [15], Wikipedia[2] [16], [17], and so on. Another work in [10] builds the hierarchical profile automatically via term-frequency analysis on the user data. In our proposed UPS framework, we do not focus on the implementation of the user profiles. Actually, our framework can potentially adopt any hierarchical representation based on a taxonomy of knowledge.

As for the performance measures of PWS in the literature, Normalized Discounted Cumulative Gain (nDCG) [18] is a common measure of the effectiveness of an information retrieval system. It is based on a human-graded relevance scale of item-positions in the result list, and is, therefore, known for its high cost in explicit feedback collection. To reduce the human involvement in performance measuring, researchers also propose other metrics of personalized web search that rely on clicking decisions, including *Average Precision (AP)* [19], [10], *Rank Scoring* [13], and *Average Rank* [3], [8]. We use the *Average Precision* metric, proposed by Dou et al. [1], to measure the effectiveness of the personalization in UPS. Meanwhile, our work is distinguished from previous studies as it also proposes two predictive metrics, namely *personalization utility* and *privacy risk*, on a profile instance without requesting for user feedback.

1. Open Directory Project (ODP), http://dmoz.org/.
2. Wikipedia, the Free Encyclopedia, http://www.wikipedia.org/.

2.2 Privacy Protection in PWS System

Generally there are two classes of privacy protection problems for PWS. One class includes those treat privacy as the identification of an individual, as described in [20]. The other includes those consider the sensitivity of the data, particularly the user profiles, exposed to the PWS server.

Typical works in the literature of protecting user identifications (class one) try to solve the privacy problem on different levels, including the *pseudoidentity*, the *group identity*, *no identity*, and *no personal information*. Solution to the first level is proved to fragile [11]. The third and fourth levels are impractical due to high cost in communication and cryptography. Therefore, the existing efforts focus on the second level. Both [21] and [22] provide online anonymity on user profiles by generating a group profile of k users. Using this approach, the linkage between the query and a single user is broken. In [23], the useless user profile (UUP) protocol is proposed to shuffle queries among a group of users who issue them. As a result any entity cannot profile a certain individual. These works assume the existence of a trustworthy third-party anonymizer, which is not readily available over the Internet at large. Viejo and Castellà-Roca [24] use legacy social networks instead of the third party to provide a distorted user profile to the web search engine. In the scheme, every user acts as a search agency of his or her neighbors. They can decide to submit the query on behalf of who issued it, or forward it to other neighbors. The shortcomings of current solutions in class one is the high cost introduced due to the collaboration and communication.

The solutions in class two do not require third-party assistance or collaborations between social network entries. In these solutions, users only trust themselves and cannot tolerate the exposure of their complete profiles an anonymity server. In [12], Krause and Horvitz employ statistical techniques to learn a probabilistic model, and then use this model to generate the near-optimal partial profile. One main limitation in this work is that it builds the user profile as a finite set of attributes, and the probabilistic model is trained through predefined frequent queries. These assumptions are impractical in the context of PWS. Xu et al. [10] proposed a privacy protection solution for PWS based on hierarchical profiles. Using a user-specified threshold, a generalized profile is obtained in effect as a rooted subtree of the complete profile. Unfortunately, this work does not address the query utility, which is crucial for the service quality of PWS. For comparison, our approach takes both the privacy requirement and the query utility into account.

A more important property that distinguishes our work from [10] is that we provide *personalized privacy protection* in PWS. The concept of personalized privacy protection is first introduced by Xiao and Tao [25] in Privacy-Preserving Data Publishing (PPDP). A person can specify the degree of privacy protection for her/his sensitive values by specifying "guarding nodes" in the taxonomy of the sensitive attribute. Motivate by this, we allow users to customize privacy needs in their hierarchical user profiles.

Aside from the above works, a couple of recent studies have raised an interesting question that concerns the privacy protection in PWS. The works in [1], [26] have

TABLE 1
Symbols and Descriptions

Symbol	Description
$\|\mathcal{T}\|$	The count of nodes of the tree $\mathcal{T}$
$t \in \mathcal{T}/N \subset \mathcal{T}$	t is a node (N is a node set) in the tree $\mathcal{T}$
$subtr(t, \mathcal{T})$	The subtree rooted on t within the tree $\mathcal{T}$
$rsbtr(N, \mathcal{T})$	The rooted subtree of $\mathcal{T}$ by removing the set N
$trie(N)$	The topic-path prefix tree built with the set N
$root(\mathcal{T})$	The root of the tree $\mathcal{T}$
$par(t, \mathcal{T})$	The parent of t in the tree $\mathcal{T}$
$lca(N, \mathcal{T})$	The least common ancestor of the set N in $\mathcal{T}$
$C(t, \mathcal{T})$	The children of t within the tree $\mathcal{T}$

found that personalization may have different effects on different queries. Queries with smaller *click-entropies*, namely distinct queries, are expected to benefit more from personalization, while those with larger values (ambiguous ones) are not. Moreover, the latter may even cause privacy disclosure. Therefore, the need for personalization becomes questionable for such queries. Teevan et al. [26] collect a set of features of the query to classify queries by their click-entropy. While these works are motivative in questioning whether to personalize or not to, they assume the availability of massive user query logs (on the server side) and user feedback. In our UPS framework, we differentiate distinct queries from ambiguous ones based on a client-side solution using the predictive query utility metric.

This paper is an extension to our preliminary study reported in [27]. In the previous work, we have proposed the prototype of UPS, together with a greedy algorithm GreedyDP (named as GreedyUtility in [27]) to support online profiling based on predictive metrics of personalization utility and privacy risk. In this paper, we extend and detail the implementation of UPS. We extend the metric of personalization utility to capture our three new observations. We also refine the evaluation model of privacy risk to support user-customized sensitivities. Moreover, we propose a new profile generalization algorithm called GreedyIL. Based on three heuristics newly added in the extention, the efficiency and stability of the new algorithm outperforms the old one significantly.

3 Preliminaries and Problem Definition

In this section, we first introduce the structure of user profile in UPS. Then, we define the customized privacy requirements on a user profile. Finally, we present the attack model and formulate the problem of privacy-preserving profile generalization. For ease of presentation, Table 1 summarizes all the symbols used in this paper.

3.1 User Profile

Consistent with many previous works in personalized web services, each user profile in UPS adopts a hierarchical structure. Moreover, our profile is constructed based on the availability of a public accessible taxonomy, denoted as $\mathcal{R}$, which satisfies the following assumption.

Assumption 1. *The repository $\mathcal{R}$ is a huge topic hierarchy covering the entire topic domain of human knowledge. That is, given any human recognizable topic t, a corresponding node (also referred to as t) can be found in $\mathcal{R}$, with the subtree $subtr(t, \mathcal{R})$ as the taxonomy accompanying t.*

The repository is regarded as publicly available and can be used by anyone as the background knowledge. Such repositories do exist in the literature, for example, the ODP [1], [14], [3], [15], Wikipedia [16], [17], WordNet [22], and so on. In addition, each topic $t \in \mathcal{R}$ is associated with a *repository support*, denoted by $sup_{\mathcal{R}}(t)$, which quantifies how often the respective topic is touched in human knowledge. If we consider each topic to be the result of a random walk from its parent topic in $\mathcal{R}$, we have the following recursive equation:

$$sup_{\mathcal{R}}(t) = \sum_{t' \in C(t,\mathcal{R})} sup_{\mathcal{R}}(t'). \quad (1)$$

Equation (1) can be used to calculate the repository support of all topics in $\mathcal{R}$, relying on the following assumption that the support values of all leaf topics in $\mathcal{R}$ are available.

Assumption 2. *Given a taxonomy repository $\mathcal{R}$, the* repository support *is provided by $\mathcal{R}$ itself for each leaf topic.*

In fact, Assumption 2 can be relaxed if the support values are not available. In such case, it is still possible to "simulate" these *repository supports* with the topological structure of $\mathcal{R}$. That is, $sup_{\mathcal{R}}(t)$ can be calculated as the count of leaves in $subtr(t, \mathcal{R})$.

Based on the taxonomy repository, we define a probability model for the topic domain of the human knowledge. In the model, the repository $\mathcal{R}$ can be viewed as a hierarchical partitioning of the *universe* (represented by the *root* topic) and every topic $t \in \mathcal{R}$ stands for a random event. The conditional probability $Pr(t \mid s)$ (s is an ancestor of t) is defined as the proportion of *repository support*:

$$Pr(t \mid s) = \frac{sup_{\mathcal{R}}(t)}{sup_{\mathcal{R}}(s)}, \quad t \in subtr(s, \mathcal{R}). \quad (2)$$

Thus, $Pr(t)$ can be further defined as

$$Pr(t) = Pr(t \mid root(\mathcal{R})), \quad (3)$$

where $root(\mathcal{R})$ is the *root* topic which has probability 1. Now, we present the formal definition of user profile.

Definition 1 (User Profile/$\mathcal{H}$). *A user profile $\mathcal{H}$, as a hierarchical representation of user interests, is a* rooted subtree *of $\mathcal{R}$. The notion* rooted subtree *is given in Definition 2.*

Definition 2 (Rooted Subtree). *Given two trees $\mathcal{S}$ and $\mathcal{T}$, $\mathcal{S}$ is a* rooted subtree *of $\mathcal{T}$ if $\mathcal{S}$ can be generated from $\mathcal{T}$ by removing a node set $X \subset \mathcal{T}$ (together with subtrees) from $\mathcal{T}$, i.e., $\mathcal{S} = rsbtr(X, \mathcal{T})$.*

A diagram of a sample user profile is illustrated in Fig. 2a, which is constructed based on the sample taxonomy repository in Fig. 2b. We can observe that the owner of this profile is mainly interested in *Computer Science* and *Music*, because the major portion of this profile is made up of fragments from taxonomies of these two topics in the sample repository. Some other taxonomies also serve in comprising the profile, for example, *Sports* and *Adults*.

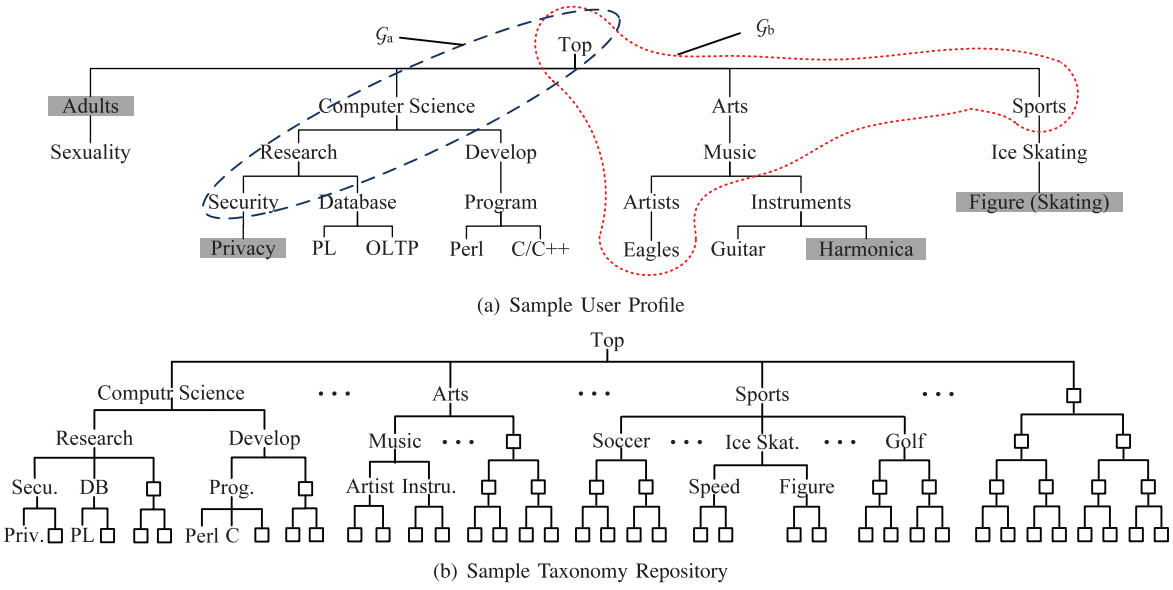


Fig. 2. Taxonomy-based user profile.

Although a user profile $\mathcal{H}$ inherits from $\mathcal{R}$ a subset of topic nodes and their links, it does not duplicate the repository supports. Instead, each topic $t \in \mathcal{H}$ is labeled with a *user support*, denoted by $sup_{\mathcal{H}}(t)$, which describes the user's preference on the respective topic t. Similar to its repository counterpart, the user support can be recursively aggregated from those specified on the leaf topics:

$$sup_{\mathcal{H}}(t) = \sum_{t' \in C(t,\mathcal{H})} sup_{\mathcal{H}}(t'). \quad (4)$$

The user support is different from the repository support as the former describes the user's preference on t, while the latter indicates the importance of t in the entire human knowledge.

3.2 Customized Privacy Requirements

Customized privacy requirements can be specified with a number of *sensitive-nodes* (topics) in the user profile, whose disclosure (to the server) introduces privacy risk to the user.

Definition 3 (SENSITIVE NODES/S). *Given a user profile $\mathcal{H}$, the* sensitive nodes *are a set of user specified sensitive topics $S \subset \mathcal{H}$, whose subtrees are* nonoverlapping, *i.e.,* $\forall s_1, s_2 \in S (s_1 \neq s_2), s_2 \notin subtr(s_1, \mathcal{H})$.

In the sample profile shown in Fig. 2a, the sensitive nodes $S = \{Adults, Privacy, Harmonica, Figure\ (Skating)\}$ are shaded in gray color in $\mathcal{H}$.

It must be noted that user's privacy concern differs from one sensitive topic to another. In the above example, the user may hesitate to share her personal interests (e.g., *Harmonica*, *Figure Skating*) only to avoid various advertisements. Thus, the user might still tolerate the exposure of such interests to trade for better personalization utility. However, the user may never allow another interest in topic *Adults* to be disclosed. To address the difference in privacy concerns, we allow the user to specify a *sensitivity* for each node $s \in S$.

Definition 4 (SENSITIVITY/$sen(s)$). *Given a* sensitive-node *s, its sensitivity, i.e., $sen(s)$, is a positive value that quantifies the severity of the privacy leakage caused by disclosing s.*

As the sensitivity values explicitly indicate the user's privacy concerns, the most straightforward privacy-preserving method is to remove subtrees rooted at all *sensitive-nodes* whose sensitivity values are greater than a threshold. Such method is referred to as *forbidding*. However, forbidding is far from enough against a more sophisticated adversary. To clearly illustrate the limitation of forbidding, we first introduce the attack model which we aim at resisting.

3.3 Attack Model

Our work aims at providing protection against a typical model of privacy attack, namely *eavesdropping*. As shown in Fig. 3, to corrupt Alice's privacy, the eavesdropper Eve successfully intercepts the communication between Alice and the PWS-server via some measures, such as *man-in-the-middle* attack, invading the server, and so on. Consequently, whenever Alice issues a query q, the entire copy of q together with a runtime profile $\mathcal{G}$ will be captured by Eve. Based on $\mathcal{G}$, Eve will attempt to touch the *sensitive nodes* of

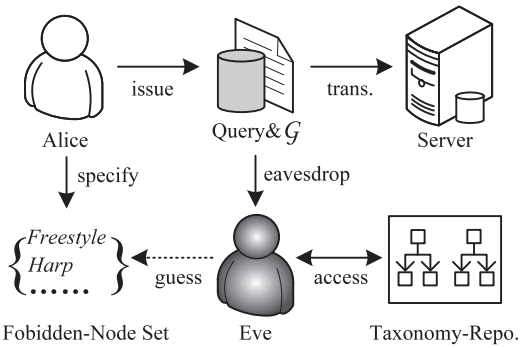


Fig. 3. Attack model of personalized web search.

Alice by recovering the segments hidden from the original $\mathcal{H}$ and computing a confidence for each recovered topic, relying on the background knowledge in the publicly available taxonomy repository $\mathcal{R}$.

Note that in our attack model, Eve is regarded as an adversary satisfying the following assumptions:

Knowledge bounded. The background knowledge of the adversary is limited to the taxonomy repository $\mathcal{R}$. Both the profile $\mathcal{H}$ and privacy are defined based on $\mathcal{R}$.

Session bounded. None of previously captured information is available for tracing the same victim in a long duration. In other words, the eavesdropping will be started and ended within a single query session.

The above assumptions seem strong, but are reasonable in practice. This is due to the fact that the majority of privacy attacks on the web are undertaken by some automatic programs for sending targeted (spam) advertisements to a large amount of PWS-users. These programs rarely act as a real person that collects prolific information of a specific victim for a long time as the latter is much more costly.

If we consider the sensitivity of each sensitive topic as the cost of recovering it, the *privacy risk* can be defined as the total (probabilistic) sensitivity of the sensitive nodes, which the adversary can probably recover from $\mathcal{G}$. For fairness among different users, we can normalize the privacy risk with $\sum_{s \in S} sen(s)$, which stands for the total wealth of the user. Our approach to privacy protection of personalized web search has to keep this privacy risk under control.

3.4 Generalizing User Profile

Now, we exemplify the inadequacy of *forbidding* operation. In the sample profile in Fig. 2a, *Figure* is specified as a sensitive node. Thus, $rsbtr(S, \mathcal{H})$ only releases its parent *Ice Skating*. Unfortunately, an adversary can recover the subtree of *Ice Skating* relying on the repository shown in Fig. 2b, where *Figure* is a main branch of *Ice Skating* besides *Speed*. If the probability of touching both branches is equal, the adversary can have 50 percent confidence on *Figure*. This may lead to high privacy risk if $sen(Figure)$ is high. A safer solution would remove node *Ice Skating* in such case for privacy protection. In contrast, it might be unnecessary to remove sensitive nodes with low sensitivity. Therefore, simply forbidding the sensitive topics does not protect the user's privacy needs precisely.

To address the problem with forbidding, we propose a technique, which detects and removes a set of nodes X from $\mathcal{H}$, such that the privacy risk introduced by exposing $\mathcal{G} = rsbtr(X, \mathcal{H})$ is always under control. Set X is typically different from S. For clarity of description, we assume that all the subtrees of $\mathcal{H}$ rooted at the nodes in X do not overlap each other. This process is called *generalization*, and the output $\mathcal{G}$ is a *generalized* profile.

The generalization technique can seemingly be conducted during offline processing without involving user queries. However, it is impractical to perform offline generalization due to two reasons:

1. The output from offline generalization may contain many topic branches, which are irrelevant to a query. A more flexible solution requires *online generalization*, which depends on the queries. Online generalization not only avoids unnecessary privacy disclosure, but also removes noisy topics that are irrelevant to the current query.

   For example, given a query $q_a =$ "K-Anonymity," which is a privacy protection technique used in data publishing, a desirable result of online generalization might be $\mathcal{G}_a$, surrounded by the dashed ellipse in Fig. 2a. For comparison, if the query is $q_b =$ "Eagles," the generalized profile would better become $\mathcal{G}_b$ contained in the dotted curve, which includes two possible intentions (one being a rock band and the other being an American football team Philadelphia Eagles). The node sets to be removed are $X_a = \{Adults, Privacy, Database, Develop, Arts, Sports\}$, and $X_b = \{Adults, Computer\ Science, Instrument, Ice\ Skating\}$, respectively.
2. It is important to monitor the personalization utility during the generalization. Using the running example, profiles $\mathcal{G}_a$ and $\mathcal{G}_b$ might be generalized to smaller rooted subtrees. However, overgeneralization may cause ambiguity in the personalization, and eventually lead to poor search results. Monitoring the utility would be possible only if we perform the generalization at runtime.

We now define the problem of privacy-preserving generalization in UPS as follows, based on two notions named *utility* and *risk*. The former measures the *personalization utility* of the generalized profile, while the latter measures the *privacy risk* of exposing the profile.

Problem 1 (δ-RISK PROFILE GENERALIZATION/δ-RPG). *Given a user profile $\mathcal{H}$ with* sensitive-nodes *S being specified, a query q, metric of privacy $risk(q, \mathcal{G})$, metric of utility $util(q, \mathcal{G})$, and a user specified threshold δ, the δ-risk profile generalization is to find an optimal instance of $\mathcal{G}$ (denoted as $\mathcal{G}*$), which satisfies*

$$\mathcal{G}* = \underset{\mathcal{G}}{\operatorname{argmax}}(util(q, \mathcal{G})), \quad risk(q, \mathcal{G}) < \delta. \tag{5}$$

In the above definition, δ represents the user's tolerance to the privacy risk (expense rate) of exposing the profile. Note that metric $risk(q, \mathcal{G})$ and $util(q, \mathcal{G})$ only depend on the instance of $\mathcal{G}$ and the query q as they are implemented to predict the privacy risk and personalization utility of $\mathcal{G}$ on q, without any user feedback. Details of these metrics will be presented in Section 5.1.

4 UPS PROCEDURES

In this section, we present the procedures carried out for each user during two different execution phases, namely the *offline* and *online* phases. Generally, the offline phase constructs the original user profile and then performs *privacy requirement customization* according to user-specified topic sensitivity. The subsequent online phase finds the Optimal δ-Risk Generalization solution in the search space determined by the customized user profile.

As mentioned in the previous section, the online generalization procedure is guided by the global risk and utility metrics. The computation of these metrics relies on two intermediate data structures, namely a *cost layer* and a *preference layer* defined on the user profile. The cost layer

defines for each node $t \in \mathcal{H}$ a cost value $cost(t) \geq 0$, which indicates the total sensitivity at risk caused by the disclosure of t. These cost values can be computed offline from the user-specified sensitivity values of the sensitive nodes. The preference layer is computed online when a query q is issued. It contains for each node $t \in \mathcal{H}$ a value indicating the user's query-related preference on topic t. These preference values are computed relying on a procedure called *query topic mapping*.

Specifically, each user has to undertake the following procedures in our solution:

1. offline profile construction,
2. offline privacy requirement customization,
3. online query-topic mapping, and
4. online generalization.

Offline-1. Profile Construction. The first step of the offline processing is to build the original user profile in a topic hierarchy $\mathcal{H}$ that reveals user interests. We assume that the user's preferences are represented in a set of plain text documents, denoted by D. To construct the profile, we take the following steps:

1. Detect the respective topic in $\mathcal{R}$ for every document $d \in D$. Thus, the preference document set D is transformed into a topic set T.
2. Construct the profile $\mathcal{H}$ as a topic-path trie with T, i.e., $\mathcal{H} = trie(T)$.
3. Initialize the user support $sup_{\mathcal{H}}(t)$ for each topic $t \in T$ with its document support from D, then compute $sup_{\mathcal{H}}(t)$ of other nodes of $\mathcal{H}$ with (4).

There is one open question in the above process—how to detect the respective topic for each document $d \in D$. In Section 6, we present our solution to this problem in our implementation.

Offline-2. Privacy Requirement Customization. This procedure first requests the user to specify a sensitive-node set $S \subset \mathcal{H}$, and the respective sensitivity value $sen(s) > 0$ for each topic $s \in S$. Next, the *cost layer* of the profile is generated by computing the cost value of each node $t \in \mathcal{H}$ as follows:

1. For each *sensitive-node*, $cost(t) = sen(t)$;
2. For each *nonsensitive* leaf node, $cost(t) = 0$;
3. For each *nonsensitive* internal node, $cost(t)$ is recursively given by (6) in a bottom-up manner:

$$cost(t) = \sum_{t' \in \mathcal{C}(t,\mathcal{H})} cost(t') \times Pr(t' \mid t). \tag{6}$$

Till now, we have obtained the customized profile with its *cost layer* available. When a query q is issued, this profile has to go through the following two online procedures:

Online-1. Query-topic Mapping. Given a query q, the purposes of query-topic mapping are 1) to compute a rooted subtree of $\mathcal{H}$, which is called a *seed profile*, so that all topics relevant to q are contained in it; and 2) to obtain the preference values between q and all topics in $\mathcal{H}$. This procedure is performed in the following steps:

1. Find the topics in $\mathcal{R}$ that are relevant to q. We develop an efficient method to compute the *relevances* of all topics in $\mathcal{R}$ with q (detailed in Section 6). These values can be used to obtain a set of *nonoverlapping* relevant topics denoted by $T(q)$, namely the *relevant set*. We require these topics to be nonoverlapping so that $T(q)$, together with all their ancestor nodes in $\mathcal{R}$, comprise a query-relevant trie denoted as $\mathcal{R}(q)$. Apparently, $T(q)$ are the leaf nodes of $\mathcal{R}(q)$. Note that $\mathcal{R}(q)$ is usually a small fraction of $\mathcal{R}$.
2. Overlap $\mathcal{R}(q)$ with $\mathcal{H}$ to obtain the *seed profile* $\mathcal{G}_0$, which is also a rooted subtree of $\mathcal{H}$. For example, by applying the mapping procedure on query "*Eagles*," we obtain a relevant set $T(Eagles)$ as shown in Table 2. Overlapping the sample profile in Fig. 2a with its query-relevant trie $\mathcal{R}(Eagles)$ gives the seed profile $\mathcal{G}_b$, whose size is significantly reduced compared to the original profile.

TABLE 2
Contents of T(Eagles)

Topics in T(*Eagles*)	Rel.
Top/Arts/Music/Artists/Eagles	23
Top/Sports/American football/NFL/Philadelphia Eagles	14
Top/Science/Biology/Animals/Birds/Raptors/Eagles	7
Top/Society/Military/Aviation/Aircraft/Fighters/F-15	4

The leaves of the seed profile $\mathcal{G}_0$ (generated from the second step) form a particularly interesting node set—the overlap between set $T(q)$ and $\mathcal{H}$. We denote it by $T_{\mathcal{H}}(q)$, and obviously we have $T_{\mathcal{H}}(q) \subset T(q)$.

Then, the preference value of a topic $t \in \mathcal{H}$ is computed as following:

1. If t is a leaf node and $t \in T_{\mathcal{H}}(q)$, its preference $pref_{\mathcal{H}}(t,q)$ is set to the long-term user support $sup_{\mathcal{H}}(q)$,[3] which can be obtained directly from the user profile.
2. If t is a leaf node and $t \notin T_{\mathcal{H}}(q)$, $pref_{\mathcal{H}}(t,q) = 0$.
3. Otherwise, t is not a leaf node. The preference value of topic t is recursively aggregated from its child topics as

$$pref_{\mathcal{H}}(t,q) = \sum_{t' \in C(t,\mathcal{H})} pref_{\mathcal{H}}(t', \mathcal{H}).$$

Finally, it is easy to obtain the *normalized preference* for each $t \in \mathcal{H}$ as

$$Pr(t \mid q, \mathcal{H}) = \frac{pref_{\mathcal{H}}(t,q)}{\sum_{t' \in T_{\mathcal{H}}(q)} pref_{\mathcal{H}}(t',q)}. \tag{7}$$

Note that the first step computes for each $t \in T(q)$ a relevance value with the query, denoted by $rel_{\mathcal{R}}(q)$. These values can be used to model a conditional probability that indicates how frequently topic t is covered by q:

$$Pr(t \mid q) = Pr(t \mid q, \mathcal{R}) = \frac{rel_{\mathcal{R}}(t,q)}{\sum_{t' \in T(q)} rel_{\mathcal{R}}(t',q)}. \tag{8}$$

3. This approximation is made based on the idea that the user's query-related preference of t can be estimated with its long-term preference in the user profile.

Though this probability is not used in this procedure, it is needed to evaluate the *discriminating power* of q (Section 5.1), and to decide whether to personalize a query or not (Section 5.2).

Online-2. Profile Generalization. This procedure generalizes the seed profile $\mathcal{G}_0$ in a cost-based iterative manner relying on the privacy and utility metrics. In addition, this procedure computes the discriminating power for online decision on whether personalization should be employed. We will elaborate these techniques in Section 5.3.

5 Generalization Techniques

In this section, we first introduce the two critical metrics for our generalization problem. Then, we present our method of online decision on personalization. Finally, we propose the generalization algorithms.

5.1 Metrics

5.1.1 Metric of Utility

The purpose of the utility metric is to predict the search quality (in revealing the user's intention) of the query q on a generalized profile $\mathcal{G}$. The reason for not measuring the search quality directly is because search quality depends largely on the implementation of PWS search engine, which is hard to predict. In addition, it is too expensive to solicit user feedback on search results. Alternatively, we transform the utility prediction problem to the estimation of the *discriminating power* of a given query q on a profile $\mathcal{G}$ under the following assumption.

Assumption 3. *When a PWS search engine is given, the search quality is only determined by the* discriminating power *of the exposed query-profile pair* $\langle q, \mathcal{G}\rangle$.

Although the same assumption has been made in [12] to model utility, the metric in that work cannot be used in our problem settings as our profile is a hierarchical structure rather than a flat one. Given a hierarchical profile $\mathcal{G}$ and a query q, we can intuitively expect more *discriminating power* when

- ob1) more specific topics are observed in $T_{\mathcal{G}}(q)$, or
- ob2) the distribution of $Pr(t \mid q, \mathcal{G})$ is more concentrated on a few topics in $T_{\mathcal{G}}(q)$, or
- ob3) the topics in $T_{\mathcal{G}}(q)$ are more similar to each other.

Therefore, an effective utility metric should be consistent with observations ob1, ob2, and ob3.

To propose our model of utility, we introduce the notion of *Information Content* (IC), which estimates how specific a given topic t is. Formally, the IC of a topic t is given by

$$IC(t) = \log^{-1} Pr(t), \tag{9}$$

where $Pr(t)$ is given in (3). The more often topic t is mentioned, the smaller IC (less specific) will it have. The *root* topic has an IC of 0, as it dominates the entire topic domain and always occurs.

Now, we develop the *first* component of the utility metric called *Profile Granularity (PG)*, which is the *KL-Divergence* between the probability distributions of the topic domain with and without $\langle q, \mathcal{G}\rangle$ exposed. That is

$$\begin{aligned} PG(q,\mathcal{G}) &= \sum_{t\in T_{\mathcal{G}}(q)} Pr(t \mid q,\mathcal{G}) \log \frac{Pr(t \mid q,\mathcal{G})}{Pr(t)} \\ &= \underbrace{\sum_{t\in T_{\mathcal{G}}(q)} Pr(t \mid q,\mathcal{G}) IC(t)}_{ob1} - \underbrace{H(t \mid q,\mathcal{G})}_{ob2}, \end{aligned} \tag{10}$$

where the probability $Pr(t \mid q, \mathcal{G})$ (referred to as *normalized preference*) can be computed with (7). We can justify that this component can capture the first two observations we proposed above, by decomposing $PG(q, \mathcal{G})$ into two terms which respect ob1 and ob2 separately. The first term can be considered as the expected IC of topics in $T_{\mathcal{G}}(q)$. The second one quantifies the uncertainty of the distribution of the user preference on topics in $T_{\mathcal{G}}(q)$. Such uncertainty is modeled as a penalty to the utility.

The second component of utility is called *Topic Similarity* (TS), which measures the semantic similarity among the topics in $T_{\mathcal{G}}(q)$ as observation ob3 suggests. This can be computed as the *Information Content* of the *Least Common Ancestor* of $T_{\mathcal{G}}(q)$ as follows:

$$TS(q,\mathcal{G}) = IC(lca(T_{\mathcal{G}}(q))). \tag{11}$$

This similarity measure was first proposed in [28], whose idea is straightforward: the more specific the common ancestor topic is, the more similarity among the topics in $T_{\mathcal{G}}(q)$.

Finally, the *discriminating power* can be expressed as a normalized combination of $PG(q, \mathcal{G})$ and $TS(q, \mathcal{G})$ as follows:

$$DP(q,\mathcal{G}) = \frac{PG(q,\mathcal{G}) + TS(q,\mathcal{G})}{2\sum_{t\in T_{\mathcal{H}}(q)} Pr(t \mid q,\mathcal{H}) IC(t)}, \tag{12}$$

where $\sum_{t\in T_{\mathcal{H}}(q)} Pr(t \mid q, \mathcal{H}) IC(t)$ is the expected IC of topics in $T_{\mathcal{H}}(q)$, given the profile $\mathcal{G}$ is generalized from $\mathcal{H}$. It is easy to demonstrate that the value of $DP(q, \mathcal{G})$ is bounded within $(0, 1]$.

Then, the *personalization utility* is defined as the gain of *discriminating power* achieved by exposing profile $\mathcal{G}$ together with query q, i.e.,

$$util(q,\mathcal{G}) = DP(q,\mathcal{G}) - DP(q,\mathcal{R}),$$

where $DP(q, \mathcal{R})$ quantifies the *discriminating power* of the query q without exposing any profile, which can be obtained by simply replacing all occurrences of $Pr(t \mid q, \mathcal{G})$ in (12) with $Pr(t \mid q)$ (obtained in (8)). Note that $util(q, \mathcal{G})$ can be negative. That is, personalization with a profile $\mathcal{G}$ may generate poorer *discriminating power*. This may happen when $\mathcal{G}$ does not reduce the uncertainty of $Pr(t \mid q)$ effectively, i.e., $T_{\mathcal{G}}(q) = T(q)$, and describes the related topics in coarser granularity.

Since $DP(q, \mathcal{R})$ is fixed whenever q is specified, the profile generalization simply take $DP(q, \mathcal{G})$ (instead of $util(q, \mathcal{G})$) to be the optimization target.

5.1.2 Metric of Privacy

The privacy risk when exposing $\mathcal{G}$ is defined as the total sensitivity contained in it, given in normalized form. In the worst case, the original profile is exposed, and the risk of exposing all sensitive nodes reaches its maximum, namely 1. However, if a sensitive node is pruned and its ancestor nodes are retained during the generalization, we still have

to evaluate the risk of exposing the ancestors. This can be done using the cost layer computed during Offline-2.

Given a generalized profile $\mathcal{G}$, the unnormalized risk of exposing it is recursively given by

$$Risk(t,\mathcal{G}) = \begin{cases} cost(t) & \text{if } t \text{ is leaf,} \\ \sum_{t' \in C(t,\mathcal{G})} Risk(t',\mathcal{G}) & \text{otherwise.} \end{cases} \quad (13)$$

However, in some cases, the cost of a nonleaf node might even be greater than the total risk aggregated from its children. For instance, in the profile $\mathcal{G}_b$ (Fig. 2a), the cost of *Music* is greater than that of *Artist* since *Music* has sensitivity propagation from its sensitive descendent *Harmonica*. Therefore, (13) might underestimate the real risk. So we amend the equation for nonleaf node as

$$Risk(t,\mathcal{G}) = max\left(cost(t), \sum_{t' \in C(t,\mathcal{G})} Risk(t',\mathcal{G})\right). \quad (14)$$

Then, the normalized risk can be obtained by dividing the unnormalized risk of the root node with the total sensitivity in $\mathcal{H}$, namely

$$risk(q,\mathcal{G}) = \frac{Risk(root,\mathcal{G})}{\sum_{s \in S} sen(s)}. \quad (15)$$

We can see that $risk(q,\mathcal{G})$ is always in the interval $[0,1]$.

5.2 Online Decision: To Personalize or Not

The results reported in [1] demonstrate that there exist a fair amount of queries called *distinct* queries, to which the profile-based personalization contributes little or even reduces the search quality, while exposing the profile to a server would for sure risk the user's privacy. To address this problem, we develop an online mechanism to decide whether to personalize a query. The basic idea is straightforward—if a *distinct* query is identified during generalization, the entire runtime profiling will be aborted and the query will be sent to the server without a user profile.

We identify distinct queries using the *discriminating power* (defined in Section 5.1). Specifically, remember that the personalization utility is defined as the gain in DP when exposing the generalized profile with the query. Thus, we consider the distinct queries as those with good DP even when the client does not expose any profile. Given a query q, if $DP(q,\mathcal{R}) \geq \mu$, where μ is a predefined threshold, then q is considered a distinct query.

The benefits of making the above runtime decision are twofold:

1. It enhances the stability of the search quality;
2. It avoids the unnecessary exposure of the user profile.

5.3 The Generalization Algorithms

We start by introducing a brute-force optimal algorithm, which is proven to be NP-hard. Then, we propose two greedy algorithms, namely the GreedyDP and GreedyIL.

5.3.1 The Brute-Force Algorithm

The brute-force algorithm exhausts all possible rooted subtrees of a given seed profile to find the optimal generalization. The privacy requirements are respected during the exhaustion. The subtree with the optimal utility is chosen as the result. Although the seed profile $\mathcal{G}_0$ is significantly smaller than $\mathcal{H}$, the exponential computational complexity of brute-force algorithm is still unacceptable. Formally, we have the following theorem whose proof is given in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2012.201.

Theorem 1. *The δ-RPG problem (Problem 1) is $\mathcal{NP}$-hard.*

5.3.2 The GreedyDP Algorithm

Given the complexity of our problem, a more practical solution would be a near-optimal greedy algorithm. As preliminary, we introduce an operator $\xrightarrow{-t}$ called *prune-leaf*, which indicates the removal of a leaf topic t from a profile. Formally, we denote by $\mathcal{G}_i \xrightarrow{-t} \mathcal{G}_{i+1}$ the process of pruning leaf t from $\mathcal{G}_i$ to obtain $\mathcal{G}_{i+1}$. Obviously, the optimal profile $\mathcal{G}*$ can be generated with a finite-length transitive closure of *prune-leaf*.

The first greedy algorithm GreedyDP works in a bottom-up manner. Starting from $\mathcal{G}_0$, in every ith iteration, GreedyDP chooses a leaf topic $t \in T_{\mathcal{G}_i}(q)$ for pruning, trying to maximize the utility of the output of the current iteration, namely $\mathcal{G}_{i+1}$. During the iterations, we also maintain a best-profile-so-far, which indicates the $\mathcal{G}_{i+1}$ having the highest discriminating power while satisfying the δ-risk constraint. The iterative process terminates when the profile is generalized to a *root*-topic. The best-profile-so-far will be the final result ($\mathcal{G}*$) of the algorithm.

The main problem of GreedyDP is that it requires recomputation of all candidate profiles (together with their discriminating power and privacy risk) generated from attempts of prune-leaf on all $t \in T_{\mathcal{G}_i}(q)$. This causes significant memory requirements and computational cost.

5.3.3 The GreedyIL Algorithm

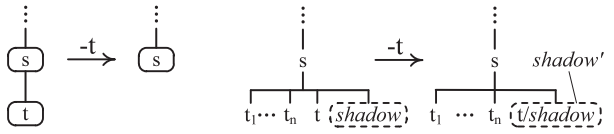
The GreedyIL algorithm improves the efficiency of the generalization using heuristics based on several findings. One important finding is that any prune-leaf operation reduces the discriminating power of the profile. In other words, the DP displays monotonicity by prune-leaf. Formally, we have the following theorem:

Theorem 2. *If $\mathcal{G}'$ is a profile obtained by applying a prune-leaf operation on $\mathcal{G}$, then $DP(q,\mathcal{G}) \geq DP(q,\mathcal{G}')$.*

Considering operation $\mathcal{G}_i \xrightarrow{-t} \mathcal{G}_{i+1}$ in the ith iteration, maximizing $DP(q,\mathcal{G}_{i+1})$ is equivalent to minimizing the incurred *information loss*, which is defined as $DP(q,\mathcal{G}_i) - DP(q,\mathcal{G}_{i+1})$.

The above finding motivates us to maintain a priority queue of candidate *prune-leaf* operators in descending order of the *information loss* caused by the operator. Specifically, each candidate operator in the queue is a tuple like $op = \langle t, IL(t,\mathcal{G}_i)\rangle$, where t is the leaf to be pruned by op and $IL(t,\mathcal{G}_i)$ indicates the IL incurred by pruning t from $\mathcal{G}_i$. This queue, denoted by $\mathcal{Q}$, enables fast retrieval of the best-so-far candidate operator.

Theorem 2 also leads to the following heuristic, which reduces the total computational cost significantly.



(a) C1: t has no siblings.

(b) C2: t has siblings $t_1 \sim t_n$. Pruning t causes the preference value of t to be shifted into that of the shadow node.

Fig. 4. Two Cases of *prune-leaf* on a leaf t.

Heuristic 1. *The iterative process can terminate whenever δ-risk is satisfied.*

The second finding is that the computation of IL can be simplified to the evaluation of $\Delta PG(q, \mathcal{G}) = PG(q, \mathcal{G}_i) - PG(q, \mathcal{G}_{i+1})$. The reason is that, referring to (12), the second term ($TS(q, \mathcal{G})$) remains unchanged for any pruning operations until a single leaf is left (in such case the only choice for pruning is the single leaf itself). Furthermore, consider two possible cases as being illustrated in Fig. 4: (C1) t is a node with no siblings, and (C2) t is a node with siblings. The case C1 is easy to handle. However, the evaluation of IL in case C2 requires introducing a *shadow* sibling[4] of t. Each time if we attempt to prune t, we actually merge t into $shadow$ to obtain a new shadow leaf $shadow'$, together with the preference of t, i.e.,

$$Pr(shadow' \mid q, \mathcal{G}) = Pr(shadow \mid q, \mathcal{G}) + Pr(t \mid q, \mathcal{G}).$$

Finally, we have the following heuristic, which significantly eases the computation of $IL(t)$. It can be seen that all terms in (16) can be computed efficiently.

Heuristic 2.

$$IL(t) = \begin{cases} Pr(t \mid q, \mathcal{G})(IC(t) - IC(par(t, \mathcal{G}))), & \text{case C1} \\ dp(t) + dp(shadow) - dp(shadow'), & \text{case C2,} \end{cases} \tag{16}$$

where $dp(t) = Pr(t \mid q, \mathcal{G}) \log \frac{Pr(t|q,\mathcal{G})}{Pr(t)}$.

The third finding is that, in case C1 described above, *prune-leaf* only operates on a single topic t. Thus, it does not impact the IL of other candidate operators in $\mathcal{Q}$. While in case C2, pruning t incurs recomputation of the preference values of its sibling nodes. Therefore, we have

Heuristic 3. *Once a leaf topic t is pruned, only the candidate operators pruning t's sibling topics need to be updated in $\mathcal{Q}$. In other words, we only need to recompute the IL values for operators attempting to prune t's sibling topics.*

Algorithm 1 shows the pseudocode of the GreedyIL algorithm. In general, GreedyIL traces the *information loss* instead of the discriminating power. This saves a lot of computational cost. In the above findings, Heuristic 1 (line 5) avoids unnecessary iterations. Heuristics 2 (line 4, 10, 14) further simplifies the computation of IL. Finally, Heuristics 3 (line 16) reduces the need for IL-recomputation significantly. In the worst case, all topics in the seed profile have sibling nodes, then GreedyIL has computational complexity of $O(|\mathcal{G}_0| * |T_{\mathcal{G}_0}(q)|)$. However, this is extremely rare in practice. Therefore, GreedyIL is expected to significantly outperform GreedyDP.

Algorithm 1: GreedyIL($\mathcal{H}$, q, δ)

Input : Seed Profile $\mathcal{G}_0$; Query q; Privacy threshold δ
Output: Generalized profile $\mathcal{G}*$ satisfying δ-Risk

1 **let** $\mathcal{Q}$ be the IL-priority queue of *prune-leaf* decisions;
  i be the iteration index, initialized to 0;
  // Online decision whether personalize q or not
2 **if** $DP(q, \mathcal{R}) < \mu$ **then**
3 Obtain the seed profile $\mathcal{G}_0$ from *Online-1*;
4 Insert $\langle t,\ IL(t)\rangle$ into $\mathcal{Q}$ for all $t \in T_{\mathcal{H}}(q)$;
5 **while** $risk(q, \mathcal{G}_i) > \delta$ **do**
6 Pop a *prune-leaf* operation on t from $\mathcal{Q}$;
7 Set $s \leftarrow par(t, \mathcal{G}_i)$;
8 Process *prune-leaf* $\mathcal{G}_i \xrightarrow{-t} \mathcal{G}_{i+1}$;
9 **if** *t has no siblings* **then** *// Case C1*
10 Insert $\langle s,\ IL(s)\rangle$ to $\mathcal{Q}$;
11 **else if** *t has siblings* **then** *// Case C2*
12 Merge t into *shadow*-sibling;
13 **if** *No operations on t's siblings in Q* **then**
14 Insert $\langle s,\ IL(s)\rangle$ to $\mathcal{Q}$;
15 **else**
16 Update the IL-values for all operations on t's siblings in $\mathcal{Q}$;
17 Update $i \leftarrow i + 1$;
18 **return** $\mathcal{G}_i$ as $\mathcal{G}*$;
19 **return** $root(\mathcal{R})$ as $\mathcal{G}*$;

6 Implementation Issues

This section presents our solutions to some of the open problems in the UPS processing. We start by introducing an inverted-indexing mechanism for computing the query-topic relevance. Then, we discuss how the topic for each document $d \in D$ is detected (Offline-1) relying on this index. Finally, we show how the query-topic relevances are computed in Online-1.

6.1 Inverted-Index of Topics

Many of the publicly available repositories allow for manual tagging and editing on each topic (e.g., DMOZ). These textual data associated with the topics comprise a document repository $D(\mathcal{R})$, so that each *leaf* topic $t \in \mathcal{R}$ finds its associated document set $D(t) \subset D(\mathcal{R})$, which describes t itself. For simplicity, we assume that $d(t_1) \bigcap d(t_2) = \phi$ if $t_1 \neq t_2$. In other words, each document in $D(\mathcal{R})$ is assigned to only one leaf topic. Thus, for each leaf topic $t \in \mathcal{R}$, it is possible to generate an inverted-index, denoted by $\mathcal{I}[t]$, containing entries like $\langle term, doc\_id, topic\_id\rangle$ for all documents in $D(t)$.

Furthermore, for each document d_i assigned to topic t (that means $d_i \in D(t)$), we also insert entries in the form of $\langle term, d_i, t\rangle$ to the index files of all the ancestor topics of t. For example, the entries of a document of *Top/Arts/Music*

4. The *shadow* sibling is dynamically generated to maintain the *nonoverlapping* property of $T_{\mathcal{G}}(q)$. The preference of *shadow* is initialized to 0. In semantics, the *shadow* stands for ANY OTHER subtopic of a topic $s \in \mathcal{G}$ apart from those presented in $C(s, \mathcal{G})$. Thus, the probability of *shadow* can always be dynamically evaluated as $Pr(shadow) = Pr(s) - \sum_{t \in C(s,\mathcal{G})} Pr(t)$.

will be inserted into $\mathcal{I}[Top/Arts/Music]$, $\mathcal{I}[Top/Arts]$ and $\mathcal{I}[Top]$.

In the end, we obtain a hierarchy of inverted indices, where each index file $\mathcal{I}[t]$ contains all the documents within the taxonomy of t. This structure enables us to efficiently process keyword search and retrieval at different semantic granularity. In particular, the root index file $\mathcal{I}[Top]$ maintains the entire document set, which can support term-based topic searching in $\mathcal{R}$.

6.2 Topic Detection in $\mathcal{R}$

During Offline-1 procedure, we need to detect the respective topic in $\mathcal{R}$ for each document $d \in D$. A naive method is to compute for each pair of d and $t \in \mathcal{R}$ their relevance with a *discriminative naive Bayesian classifier* as defined in [29]:

$$dnb(d,t) = \sum_{w \in d} N_{d,w} \ln \frac{N_{t,w} + \epsilon}{\sum_{t' \in \mathcal{R}} N_{t',w} + \epsilon}, \tag{17}$$

where $N_{t,w}$ is the frequency of word w in topic t, $N_{d,w}$[5] is the frequency of w in d, and ϵ is a smoothing factor. The topic with the largest dnb value is considered the result. Note that in (17), the values of $N_{d,w}$, $N_{t,w}$, and $N_{t',w}$ can all be obtained from the hierarchical inverted indices proposed in Section 6.1. Unfortunately, the naive method is inefficient as many of the topics in $\mathcal{R}$ are not relevant to the documents in D.

A more efficient way (and the one used in our implementation) is to exploit the user's click log to be the set D. The click log contains entries like $\langle q_i, d_1^i, d_2^i, \ldots \rangle$, where q_i is the ith query in the log and d_j^i is the jth document clicked by the user after issuing q_i. Note that $d_j^i \in D$. Thus, we can reduce the need for computation of (17) to the topics which are retrieved by q_i from the topmost inverted index $\mathcal{I}[Top]$. Specifically, for q_i and $d_j^i \in D$, we retrieve all documents relevant to q_i from the inverted index and obtain their associated topics (from the $topic\_id$). These topics are denoted by $T(q_i)$. Then, the dnb value for each $t \in T(q_i)$ is computed as

$$dnb(d,t) = \sum_{w \in d} N_{d,w} \ln \frac{N_{t,w} + \epsilon}{\sum_{t' \in T(q_i)} N_{t',w} + \epsilon}. \tag{18}$$

6.3 Query-Topic Relevance

The computation of query-topic relevance during Online-1 is straightforward. Given a query q, we retrieve from inverted index $\mathcal{I}[Top]$ the documents relevant to q using the conventional approach. These documents are then grouped by their respective topics. The relevance of each topic is then computed as the number of documents contained in each topic.

We note that the relevance metric used in our implementation is very simple and fast to evaluate. It can easily be replaced by more complicated versions.

7 Experimental Results

In this section, we present the experimental results of UPS. We conduct four experiments on UPS. In the first experiment, we study the detailed results of the metrics in each iteration of the proposed algorithms. Second, we look at the effectiveness of the proposed *query-topic mapping*. Third, we study the scalability of the proposed algorithms in terms of response time. In the fourth experiment, we study the effectiveness of *clarity prediction* and the search quality of UPS.

7.1 Experimental Setup

The UPS framework is implemented on a PC with a Pentium Dual-Core 2.50-GHz CPU and 2-GB main memory, running Microsoft Windows XP. All the algorithms are implemented in Java.

The topic repository uses the ODP web Directory. To focus on the pure English categories, we filter out taxonomies "*Top/World*" and "*Top/Adult/World.*" The click logs are downloaded from the online AOL query log, which is the most recently published data we could find. The AOL query data contain over 20 million queries and 30 million clicks of 650k users over 3 months (March 1, 2006 to May 31, 2006). The data format of each record is as follows:

$$\langle uid, query, time[, rank, url] \rangle,$$

where the first three fields indicate user *uid* issued *query* at timestamp *time*, and the last two optional fields appear when the user further clicks the *url* ranked at position *rank* in the returned results.

The profiles used in our experiment can be either *synthetic* or generated from *real query logs*:

- *Synthetic*. We cluster all AOL queries by their DP into three groups using the 1-dimensional k-means algorithm. These three groups, namely *Distinct Queries*, *Medium Queries*, and *Ambiguous Queries*, can be specified according to the following empirical rules obtained by splitting the boundaries between two neighboring clusters.
  - *Distinct Queries* for $DP(q,\mathcal{R}) \in (0.82, 1]$.
  - *Medium Queries* for $DP(q,\mathcal{R}) \in (0.44, 0.82)$.
  - *Ambiguous Queries* for $DP(q,\mathcal{R}) \in (0, 0.44)$.

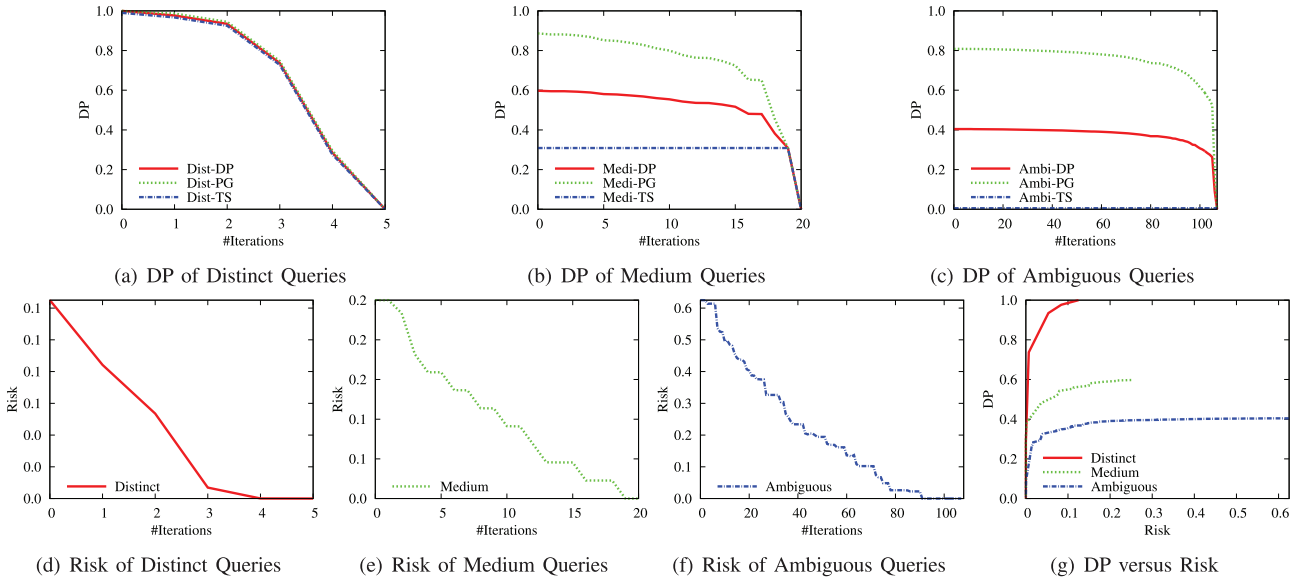
  Each synthetic profile is built from the click log of three queries, with one from each group. The forbidden node set S is selected randomly from the topics associated with the clicked documents.
- *Real*. The real user profiles are extracted from 50 distinct user click logs (with #clicks $\geq 2{,}000$) from AOL. For each user, the user profile is built with the documents dumped from all $urls$ in his/her log.[6] The sensitive nodes are randomly chosen from no more than five topics (with depth ≥ 3).

7.2 Ex1: Micro Results of Queries

In this experiment, we analyze and compare the effect of the generalization on queries with different *discriminating*

5. $N_{d,w}$ is not necessary to count, since $dnb(d,t)$ can be obtained by incrementally increasing the value by $\ln \frac{N_{t,w}+\epsilon}{\sum_{t' \in T(q)} N_{t',w}+\epsilon}$ for $w \in d$.

6. In the original AOL-data, the url field is truncated to the domain name (e.g., www.une.edu/mwwc/ becomes www.une.edu) except the URL is of the protocol *https*. As a result, we have to recover the full URLs by means of *REPLAY*. That is, to reissue the *query* of the AOL-records to some designate search engines and then retrieve the URLs matching *url*. We processed *REPLAY*, respectively, on the top 100 results returned by Yahoo and ODP, and 40 percent (#clicks ≥ 800) and 25 percent (#clicks ≥ 500) full URLs are recovered.



(a) DP of Distinct Queries (b) DP of Medium Queries (c) DP of Ambiguous Queries

(d) Risk of Distinct Queries (e) Risk of Medium Queries (f) Risk of Ambiguous Queries (g) DP versus Risk

Fig. 5. Results of Distinct/Medium/Ambiguous queries during each iteration in GreedyDP/GreedyIL. All results are obtained from the same profile.

power, and study the tradeoff between the utility and the privacy risk in the GreedyDP/GreedyIL algorithm. To clearly illustrate the difference between the three groups of queries, we use the synthetic profiles in this experiment. We perform iterative generalization on the profile using one of the original queries for creating the profile itself. The DP and risk are measured after each iteration. As the results of different profiles display similar trends, we only plot the results of three representative queries ("*Wikipedia*" for distinct queries, "*Freestyle*" for medium queries, and "*Program*" for ambiguous queries) in Fig. 5.

As Figs. 5a, 5b, and 5c show the *discriminating power* of all three sample queries displays a diminishing-returns property during generalization, especially the ambiguous one (i.e., "Program"). This indicates that the higher-level topics in the profile are more effective in improving the search quality during the personalization, while the lower-level ones are less. This property has also been reported in [12], [10]. In addition, we also plot the results of *profile granularity* and *topic similarity* (TS) across iterations in these figures. We observe that for all three samples, 1) PG shows an exactly similar trend as that of DP, 2) TS remains unchanged until the last few iterations of generalization. In particular, the TS of the ambiguous one is always 0. The reason of such results is that TS is fixed before the generalization reaches the least common ancestor of the related queries, which means PG shapes the overall DP more.

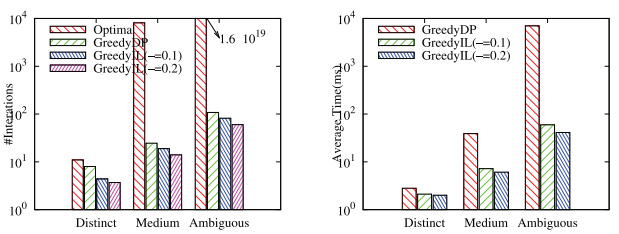
Similarly, Figs. 5d, 5e, and 5f show the results of risk during the generalization. The value of the metric first declines rapidly, but the decrease slows down as more specific profile information becomes hidden. Fig. 5g illustrates the tradeoff pattern of DP versus risk of three sample queries. For all queries, we observe an apparent "knee" on their tradeoff curve. Before this turning point, small concessions on risk can bring great promotion on utility; while after that, any tiny increase of utility will lead to enormous increase in risk. Hence, the knee is a near-optimal point for the tradeoff. We also find that the knee can be reached within limited iterations for all cases (when risk is below 0.1).

7.3 Ex2: Efficiency of Generalization Algorithms

To study the efficiency of the proposed generalization algorithms, we perform GreedyDP and GreedyIL algorithms on *real* profiles. The queries are randomly selected from their respective query log. We present the results in terms of average number of iterations and the response time of the generalization.

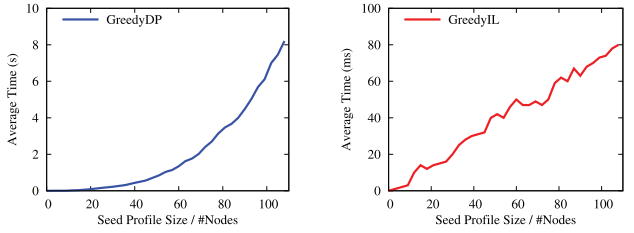
Fig. 6 shows the results of the experiment. For comparison, we also plot the theoretical number of iterations of the *Optimal* algorithm. It can be seen that both greedy algorithm outperform Optimal. GreedyDP bounds the search space to the finite-length transitive closure of *prune-leaf*. GreedyIL further reduces this measure with Heuristic 1. The greater the privacy threshold δ, the fewer iterations the algorithm requires.

The advantage of GreedyIL over GreedyDP is more obvious in terms of response time, as Fig. 6b shows. This is because GreedyDP requires much more recomputation of DP, which incurs lots of logarithmic operations. The problem worsens as the query becomes more ambiguous. For instance, the average time to process GreedyDP for queries in the ambiguous group is more than 7 seconds. In contrast, GreedyIL incurs a much smaller real-time cost, and outperforms GreedyDP by two orders of magnitude.



(a) Results of Average #Iterations (b) Results of Average Response Time

Fig. 6. Efficiency of Optimal/GreedyDP/GreedyIL.



(a) Results of GreedyDP (b) Results of GreedyIL

Fig. 7. Scalability by varying profile size.

7.4 Ex3: Scalability of Generalization Algorithms

We study the scalability of the proposed algorithms by varying 1) the seed profile size (i.e., number of nodes), and 2) the data set size (i.e., number of queries). For each possible seed profile size (ranging from 1 to 108), we randomly choose 100 queries from the AOL query log, and take their respective $\mathcal{R}(q)$ as their seed profiles. All leaf nodes in a same seed profile are given equal user preference. These queries are then processed using the GreedyDP and GreedyIL algorithms. For fair comparison, we set the privacy threshold $\delta = 0$ for GreedyIL to make it always run the same number of iterations as GreedyDP does. Fig. 7 shows the average response time of the two algorithms while varying the seed profile size. It can be seen that the cost of GreedyDP grows exponentially, and exceeds 8 seconds when the profile contains more than 100 nodes. However, GreedyIL displays near-linear scalability, and significantly outperforms GreedyDP.

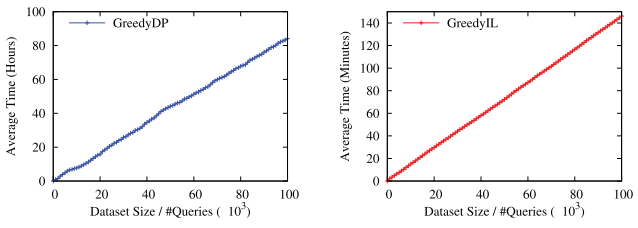
Fig. 8 illustrates the results of data sets containing different numbers of queries (from 1,000 to 100,000 queries). Apparently both algorithms have linear scalability by the data set size. For the largest data set containing 100,000 queries, it took GreedyDP 84 hours to complete all queries while GreedyIL less than 150 minutes.

7.5 Ex4: Effective Analysis of Personalization

In this experiment, we evaluate the real search quality on commercial search engines using our UPS framework. The search results is reranked with the generalized profile output by GreedyIL over 50 target users. The final search quality is evaluated using the Average Precision of the click records of the users, which is defined as

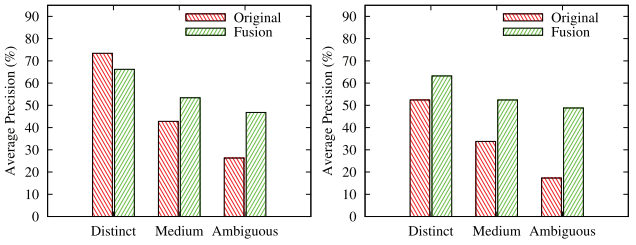
$$\mathcal{AP} = \sum_{i=1}^{n} \frac{i}{l_i.rank}/n, \tag{19}$$

where l_i is the ith relevant link identified for a query, and n is the number of relevant links.



(a) Results of GreedyDP (b) Results of GreedyIL

Fig. 8. Scalability by varying data set size.



(a) Yahoo (b) ODP

Fig. 9. Effectiveness of personalization on test queries.

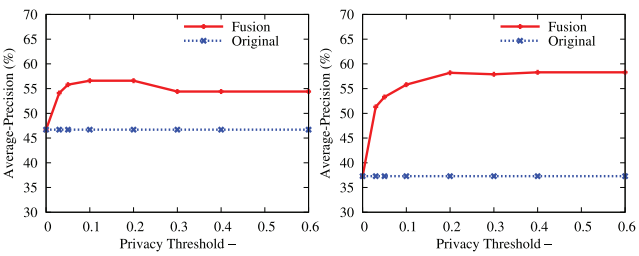
For each test query, the framework computes the final personalized rank as the Borda fusion [1] of the *UPRank* and the original rank, and then evaluate $\mathcal{AP}$ of the search results on both the fusion and the original rank. UPRank is achieved by sorting link items l in the descending order of $uscore$, which is the weighted sum over related topics in profile $\mathcal{G}*$, where the weight $dnb(l,t)$ is the relevance quantified in (17). The $uscore$ is given by

$$uscore(l) = \sum_{t \in T_{\mathcal{G}*}(q)} dnb(l,t). \tag{20}$$

Fig. 9 shows the average $\mathcal{AP}$ of the ranks before (Original) and after (Fusion) personalizing the test queries on Yahoo and ODP, respectively. The GreedyIL has a $\delta = 0.1$ and online decision mechanism disabled. From the results of both search engines, we can observe that improvements of the search quality for Medium Queries and Ambiguous Queries are much more significant than that of Distinct Queries. In particular, the personalization on Distinct Queries of Yahoo results reduces the average performance from 73.4 to 66.2 percent. This is because some irrelevant profile topics (noises) are added. The results demonstrate that profile-based personalization is more suitable for queries with small $DP(q,\mathcal{R})$.

Fig. 10 shows the results of search quality by varying the δ threshold. It is observed that the average precision of FusionRank increases rapidly when δ grows from 0.0 to 0.1. Then, further increasing δ (in effect exposing more specific topics) will only improve the search quality marginally. Moreover, the $\mathcal{AP}$ of FusionRank based on Yahoo (Fig. 10a) has a significant drop when $\delta >= 0.3$.

A comparison between the personalization results of ODP and Yahoo reveal that, although the original ODP-Rank ($\mathcal{AP} = 37.3\%$) is poorer than the original Yahoo-Rank ($\mathcal{AP} = 46.7\%$), personalization on ODP will generate better ranking than that on Yahoo. The reason for this



(a) Yahoo (b) ODP

Fig. 10. Effectiveness of personalization on varying δ.

may be that the document-distribution of ODP over all the available topics is expectedly more consistent with its own taxonomy repository, which has been employed in our implementation.

8 Conclusions

This paper presented a client-side privacy protection framework called UPS for personalized web search. UPS could potentially be adopted by any PWS that captures user profiles in a hierarchical taxonomy. The framework allowed users to specify customized privacy requirements via the hierarchical profiles. In addition, UPS also performed online generalization on user profiles to protect the personal privacy without compromising the search quality. We proposed two greedy algorithms, namely GreedyDP and GreedyIL, for the online generalization. Our experimental results revealed that UPS could achieve quality search results while preserving user's customized privacy requirements. The results also confirmed the effectiveness and efficiency of our solution.

For future work, we will try to resist adversaries with broader background knowledge, such as richer relationship among topics (e.g., exclusiveness, sequentiality, and so on), or capability to capture a series of queries (relaxing the second constraint of the adversary in Section 3.3) from the victim. We will also seek more sophisticated method to build the user profile, and better metrics to predict the performance (especially the utility) of UPS.

Acknowledgments

This work was supported by the National Science Foundation of China (No. 61170034), and China Key Technology R&D Program (No. 2011BAG05B04).

References

[1] Z. Dou, R. Song, and J.-R. Wen, "A Large-Scale Evaluation and Analysis of Personalized Search Strategies," *Proc. Int'l Conf. World Wide Web (WWW)*, pp. 581-590, 2007.

[2] J. Teevan, S.T. Dumais, and E. Horvitz, "Personalizing Search via Automated Analysis of Interests and Activities," *Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, pp. 449-456, 2005.

[3] M. Spertta and S. Gach, "Personalizing Search Based on User Search Histories," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence (WI)*, 2005.

[4] B. Tan, X. Shen, and C. Zhai, "Mining Long-Term Search History to Improve Search Accuracy," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, 2006.

[5] K. Sugiyama, K. Hatano, and M. Yoshikawa, "Adaptive Web Search Based on User Profile Constructed without any Effort from Users," *Proc. 13th Int'l Conf. World Wide Web (WWW)*, 2004.

[6] X. Shen, B. Tan, and C. Zhai, "Implicit User Modeling for Personalized Search," *Proc. 14th ACM Int'l Conf. Information and Knowledge Management (CIKM)*, 2005.

[7] X. Shen, B. Tan, and C. Zhai, "Context-Sensitive Information Retrieval Using Implicit Feedback," *Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development Information Retrieval (SIGIR)*, 2005.

[8] F. Qiu and J. Cho, "Automatic Identification of User Interest for Personalized Search," *Proc. 15th Int'l Conf. World Wide Web (WWW)*, pp. 727-736, 2006.

[9] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel, "Personalized Search," *Comm. ACM*, vol. 45, no. 9, pp. 50-55, 2002.

[10] Y. Xu, K. Wang, B. Zhang, and Z. Chen, "Privacy-Enhancing Personalized Web Search," *Proc. 16th Int'l Conf. World Wide Web (WWW)*, pp. 591-600, 2007.

[11] K. Hafner, *Researchers Yearn to Use AOL Logs, but They Hesitate*, New York Times, Aug. 2006.

[12] A. Krause and E. Horvitz, "A Utility-Theoretic Approach to Privacy in Online Services," *J. Artificial Intelligence Research*, vol. 39, pp. 633-662, 2010.

[13] J.S. Breese, D. Heckerman, and C.M. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Conf. Uncertainty in Artificial Intelligence (UAI)*, pp. 43-52, 1998.

[14] P.A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter, "Using ODP Metadata to Personalize Search," *Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development Information Retrieval (SIGIR)*, 2005.

[15] A. Pretschner and S. Gauch, "Ontology-Based Personalized Search and Browsing," *Proc. IEEE 11th Int'l Conf. Tools with Artificial Intelligence (ICTAI '99)*, 1999.

[16] E. Gabrilovich and S. Markovich, "Overcoming the Brittleness Bottleneck Using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge," *Proc. 21st Nat'l Conf. Artificial Intelligence (AAAI)*, 2006.

[17] K. Ramanathan, J. Giraudi, and A. Gupta, "Creating Hierarchical User Profiles Using Wikipedia," *HP Labs*, 2008.

[18] K. Järvelin and J. Kekäläinen, "IR Evaluation Methods for Retrieving Highly Relevant Documents," *Proc. 23rd Ann. Int'l ACM SIGIR Conf. Research and Development Information Retrieval (SIGIR)*, pp. 41-48, 2000.

[19] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley Longman, 1999.

[20] X. Shen, B. Tan, and C. Zhai, "Privacy Protection in Personalized Search," *SIGIR Forum*, vol. 41, no. 1, pp. 4-17, 2007.

[21] Y. Xu, K. Wang, G. Yang, and A.W.-C. Fu, "Online Anonymity for Personalized Web Services," *Proc. 18th ACM Conf. Information and Knowledge Management (CIKM)*, pp. 1497-1500, 2009.

[22] Y. Zhu, L. Xiong, and C. Verdery, "Anonymizing User Profiles for Personalized Web Search," *Proc. 19th Int'l Conf. World Wide Web (WWW)*, pp. 1225-1226, 2010.

[23] J. Castellí-Roca, A. Viejo, and J. Herrera-Joancomartí, "Preserving User's Privacy in Web Search Engines," *Computer Comm.*, vol. 32, no. 13/14, pp. 1541-1551, 2009.

[24] A. Viejo and J. Castellà-Roca, "Using Social Networks to Distort Users' Profiles Generated by Web Search Engines," *Computer Networks*, vol. 54, no. 9, pp. 1343-1357, 2010.

[25] X. Xiao and Y. Tao, "Personalized Privacy Preservation," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD)*, 2006.

[26] J. Teevan, S.T. Dumais, and D.J. Liebling, "To Personalize or Not to Personalize: Modeling Queries with Variation in User Intent," *Proc. 31st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, pp. 163-170, 2008.

[27] G. Chen, H. Bai, L. Shou, K. Chen, and Y. Gao, "Ups: Efficient Privacy Protection in Personalized Web Search," *Proc. 34th Int'l ACM SIGIR Conf. Research and Development in Information*, pp. 615-624, 2011.

[28] J. Conrath, "Semantic Similarity based on Corpus Statistics and Lexical Taxonomy," *Proc. Int'l Conf. Research Computational Linguistics (ROCLING X)*, 1997.

[29] D. Xing, G.-R. Xue, Q. Yang, and Y. Yu, "Deep Classifier: Automatically Categorizing Search Results into Large-Scale Hierarchies," *Proc. Int'l Conf. Web Search and Data Mining (WSDM)*, pp. 139-148, 2008.

Lidan Shou received the PhD degree in computer science from the National University of Singapore. He is an associate professor with the College of Computer Science, Zhejiang University, China. Prior to joining the faculty, he had worked in the software industry for more than two years. His research interests include spatial database, data access methods, visual and multimedia databases, and web data mining. He is a member of the ACM.

He Bai received the BS degree in computer science from Zhejiang University in 2006, and is currently working toward the PhD degree in the College of Computer Science, Zhejiang University. His research interests include data privacy protection, text data mining, and personalized information retrieval.

Ke Chen received the PhD degree in computer science in 2007, and later became a postdoctoral fellow at the School of Aeronautics and Astronautics of Zhejiang University until year 2009. She is an associate professor at the College of Computer Science, Zhejiang University. Her research interests include spatial temporal data management, web data mining, and data privacy protection. She is a member of the ACM.

Gang Chen received the PhD degree in computer science from Zhejiang University. He is a professor in the College of Computer Science and the director of the Database Lab, Zhejiang University. He has successfully led the investigation in research projects which aim at building China's indigenous database management systems. His research interests range from relational database systems to large-scale data management technologies supporting massive Internet users. He is a member of the ACM and a senior member of China Computer Federation.