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

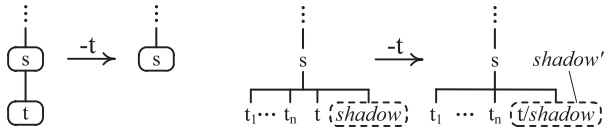
The GreedyIL algorithm improves the efficiency of the generalization using heuristics based on several findings. One important finding is that any prune-leaf operation reduces the discriminating power of the profile. In other words, the DP displays monotonicity by prune-leaf. Formally, we have the following 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 $Pr(shadow) = Pr(s) - \sum_{t \in C(s,\mathcal{G})} Pr(t)$.