Data Min Knowl Disc (2017) 31:1419–1443
DOI 10.1007/s10618-017-0524-z

The best privacy defense is a good privacy offense: obfuscating a search engine user's profile

Jörg Wicker[1] · Stefan Kramer[1]

Received: 4 December 2016 / Accepted: 19 June 2017 / Published online: 28 June 2017

Abstract User privacy on the internet is an important and unsolved problem. So far, no sufficient and comprehensive solution has been proposed that helps a user to protect his or her privacy while using the internet. Data are collected and assembled by numerous service providers. Solutions so far focused on the side of the service providers to store encrypted or transformed data that can be still used for analysis. This has a major flaw, as it relies on the service providers to do this. The user has no chance of actively protecting his or her privacy. In this work, we suggest a new approach, empowering the user to take advantage of the same tool the other side has, namely data mining to produce data which obfuscates the user's profile. We apply this approach to search engine queries and use feedback of the search engines in terms of personalized advertisements in an algorithm similar to reinforcement learning to generate new queries potentially confusing the search engine. We evaluated the approach using a real-world data set. While evaluation is hard, we achieve results that indicate that it is possible to influence the user's profile that the search engine generates. This shows that it is feasible to defend a user's privacy from a new and more practical perspective.

Responsible editors: Kurt Driessens, Dragi Kocev, Marko Robnik Šikonja, Myra Spiliopoulou

Electronic supplementary material The online version of this article (doi:10.1007/s10618-017-0524-z) contains supplementary material, which is available to authorized users.

✉ Jörg Wicker
wicker@uni-mainz.de

Stefan Kramer
kramer@informatik.uni-mainz.de

[1] Institute of Computer Science, Johannes Gutenberg University Mainz, Staudingerweg 9, 55128 Mainz, Germany

Keywords Privacy · Search engines · Personalized ads · Web mining · Reinforcement learning

1 Introduction

The Internet has become a big part of life for the majority of people in most countries of the world. Reading news, shopping, communication, listening to music, or watching movies have all been transformed from being offline activities to activities carried out on the Internet. Every part of a person's life can involve the Internet. While this is a great achievement and also simplifies many tasks, this development also brings about problems. One of them, which is also a focus of much research in data mining (Agrawal and Srikant 2000), is the question of the privacy of users. It is still not solved how to allow the service providers to store information on the user, which is what they need to do to provide the services, and at the same time encode data in a way that the provider can still use, without the users being exploited.

Additionally, privacy preserving techniques have some major design flaws: First, they rely on providers to process the data, in other words, the user has to trust the provider and has no actual control. Second, providers do not have any incentive for using such techniques. There is no advantage for them: they even have to process data before storage, which incurs costs. Third, in some cases, it is not possible to modify the data, as their sole function is to share data with contacts. Therefore, most service providers on the internet do not use privacy preserving technologies, in fact, most research on privacy preserving data mining addresses data sharing, not the problem that a single service provider stores too much data on a person (Aldeen et al. 2015; Xu et al. 2014). The user, on the other hand, does not have a choice and cannot avoid using certain service providers. For example, in the case of search engines, the user has no choice and needs them to find information on the Internet. While usually no user account is required to use search engines, the search engine can and does store information on the user, for instance, identified via cookies. Furthermore, browser fingerprints could, in theory, be used to identify users (Eckersley 2010; Nikiforakis et al. 2013). Clearly, the search engine has the right to store information on users, if stated in the terms and conditions. Nevertheless, the user usually reveals a vast amount of private data to the search engines, which can then easily be used to generate detailed profiles. For example, in 2006 AOL released the search history of over 650,000 users for a short time for research purposes. While the users were not identified via name, in many cases their search history could be used to identify the person.[1] A study in Purcell et al. (2012) found that the majority of search engine users view online targeted advertisement negatively, even 73% of users say collecting user information to personalize search results is not okay. This shows that a vast amount of users would prefer that the search engine does not collect data on them and new forms of privacy preserving are sought.

[1] While AOL retracted the data, several pages still provide access to the data and keep analyzing it, e.g., see http://www.aolstalker.com/.

In this paper, we target privacy from another perspective: We suggest to give the user tools to defend her or his privacy, such that the user does not have to rely on the other (in some cases uncontrollable) side. Data is usually stored in massive amounts and analyzed automatically using data mining techniques. The obvious method to hide the user's identity would be to flood the data storage with random data and hope the user's interest or identity would be obfuscated. However, data mining algorithms are designed to and can quite well distinguish signal from random noise. Hence, this approach would not work in most settings where the data is analyzed with sophisticated data mining algorithms.

We therefore propose to give the user the same set of tools as the other side uses to identify and learn about the user, namely machine learning and data mining techniques. In some cases, for example, in the case of hackers or government agencies, it is hard for the user to know what the other side knows about her or him, and thus hard to use techniques that hide her or his identity. In many other cases, however, the user gets some kind of feedback from the other side. In this paper, we use a widely used application as an example, namely search engines and their feedback given by personalized advertisements (see for example Bilenko and Richardson 2011).

Search engines have a huge incentive to know as much a possible about a user. First of all, personalized results can be shown to the user, improving the user's experience. However, more important is the display of personalized ads to the user. Every search engine shows personalized ads, each ad display tailored to fit the interest of the user in the long term, as well as the search query. This is a great advantage for the advertisers, as they can be sure their ads are shown to interested users. However, this also gives the possibility to the user to find out what the search engine knows about her or him.

In this paper, we present and evaluate a method that uses feedback from search engines in terms of personalized ads, to generate queries that are submitted back to the search engine to obfuscate the user's identity. Overall, there are three main contributions of this paper:

- We introduce a new approach closely related to adversarial data mining (Lowd and Meek 2005) that utilizes data mining and machine learning as a method to defend a user's privacy.
- The new approach is explored in a common setting, i.e., the use of Internet search engines.
- A learning method is presented for environments where a user can get feedback from her or his counterpart.

The remainder of this paper is organized as follows: First, we give a brief discussion and define the user and search engine model. Next, we introduce the proposed method. Then, we give details of the experimental set-up and present experimental results. Subsequently, we summarize related work before we give conclusions.

2 Discussion

This paper introduces a first proposal and study of an algorithm that targets a sophisticated data mining algorithm that exploits users' data, while at the same time not giving direct feedback. It should be understood as proof-of-concept and not a final product ready for the market. To evaluate if the general approach works, we make a number of simplifying assumptions on the user and search engine model. It is a reasonable strategy to start with the investigation of a simple setting and only then move forward to more complex settings. However, even with these simple assumptions, we can show that it is indeed possible to guide the reaction of a search engine into an intended direction.

2.1 User model and use case

The first major simplification in the user model is the reduction of the interest category for each user to one. In reality, users are interested in a variety of topics and submit queries from each of these topic categories. However, to simplify the anyway hard target of getting a reaction from a search engine, we reduced each user to one defined interest category at a time. This also simplifies the evaluation, as we can compare the reaction to one interest category. Future work will address users with multiple interest categories and users with variably strong interest in multiple categories.

Due to the nature of the problem, in our evaluation, we only addressed query results that provided ads. However, not all queries result in ads. A result without ads might be due to various reasons, for example, a user categorization by the search engine that does not align with any advertiser, or simply no ads provider wishes to advertise at that time towards this kind of user. This does provide some information that could be used in the algorithm, however, it is also left for future work to simplify the target.

Even with these limitations in the user model, there is a use case for the algorithm. A user could use a browser extension that, in the background, from time to time submits queries to the search engine and the results of the generated and real queries of the user are used in the plugin. The training data could be potentially shared among multiple users. Using this, the user could follow a normal behavior without strong modifications, use all the benefits of being identified as a certain user, but will not be recognized correctly by the search engine and would have a different personality in its perspective. To emulate this use case, in our experiments we set a "true browsing ratio" θ to a fixed value. That is, in a θ fraction of cases, the users submit a query from their own real interest category, in the remaining cases, they submit a query suggested by the corresponding algorithm.

The complexity of the presented approach is rather low. Users could use a browser plugin or app on the phone without any training. The generation process of new queries is done by a simple prediction of an online model, which does not need a lot of resources. The training process is also carried out by an online learner. Online learners are optimized to efficiently process large amounts of data using only few resources. In the presented scenario, the data is rather small. The model could be updated on a per user basis, or, alternatively centrally on a server collecting user queries and responses.

2.2 Search engine model and decision surface

Similar to the user model, we imposed some simplifications on the search engine's model. In general, the user's categorization is identified by predicting probabilities for categories, hence the predicted assignment of a user is a distribution of probabilities over the categories. Of course, this relies only on a snapshot that reflects the last query of the user, which might not always return the real assignment of the user. The ad slots are usually given to advertisers relying on a complicated bidding system, the displayed ad depends on the day, the time, the user profile, the query, and many other things. This paper tries to avoid these aspects by using a large number of users from multiple locations that use the search engine over a long period of time in a defined time slot per day. Due to the long time period and vast number of queries, the effect of aspects not considered in the search engine model should be reduced.

As the process of ad selection for a user is a complicated process, depending on many aspects, the used model is a strong simplification of the search engine model. Additionally, a search engine uses a lot more information than the submitted queries to categorize the user. However, our approach does not provide the search engine with any further information, it has to rely on the categorization based on the submitted queries. Furthermore, the results of our experiments (see Sect. 5) show that we can indeed get some signal from analyzing the ads, not just random noise. This means that despite strong limitations, we can use this approach in a meaningful way. Modern search engines use the user categorization not only for personalized advertisements, but also for personalized search results. Clearly, there is a deeper underlying trade-off between privacy and personalization. In this paper, we assume the user prefers privacy over personalization in this trade-off. Note, however, that the strength of this preference can in principle be parameterized according to the above true browsing ratio.

While most approaches that use a similar method rely on the assumption that all users behave honestly and fail the moment a large enough fraction of users use the method, this is not the case in this paper. As the suggested algorithm adapts based on the reaction of the search engine, it would find new ways of avoiding search engine identification the moment the search engine adapts to its behavior. The only effective countermeasures would be not to use personalized ads on these users, hence making user categorization useless for them.

3 Method

To define the problem, let T be the tree (hierarchy) of user interest categories, where each node V_j in this tree is annotated with a specific user interest category κ_j (an example of such a hierarchy is given in Fig. 2). The union of all such categories is given by set K. Furthermore, let $d_T(\kappa_j, \kappa_k)$ be the tree distance between κ_j and κ_k, i.e., the number of edges between the two nodes in the tree, and κ_i be the main and only interest category of a user. Then, for each category κ_j, we define Q_j as a set of example queries of category κ_j, i.e., queries that users interested in the topic of κ_j would typically send to the search engine. The response to such a query $q_l \in Q_j$ is a set of ads A_l returned by the search engine.

As full knowledge of the ground truth (i.e., the state and information of the search engine) is not available, but only feedback from the search engine in the form of ads, we define our objective function σ as follows:

$$\sigma(\kappa_i, P) = \sum_{p_j \in P, \kappa_j \in T} p_j d_T(\kappa_i, \kappa_j). \quad (1)$$

This function weights the distance between the interest category and any other category by the estimated probability of the query belonging to that category. It sums over all such weighted distances in the tree.[2] While this is the objective function we aim to minimize, we are also interested in a low probability p_j of A_j belonging to the main user interest category κ_i in the below evaluation (with P being the vector of all probabilities p_j for all categories $\kappa_j \in T$). p can be estimated by any classifier. In our case, we used text mining-based classifiers f_i for each category κ_i.

Query q_j is chosen by applying an action $a(\kappa_{in}) \rightarrow \kappa_{out}$ to the category of the previous query q_{j-1}. An action is chosen by predicting the potential score σ using online classifiers g_i, one for each action a_i, that are updated every time an action is applied. The state is given by what the search engine stored about the user, information of that is revealed by the ads and therefore the predictions $p_j \in P$ of classifiers f_j. We implemented various actions, described in detail in Sect. 3.3. The score function is given by σ and directly used in the online learners f_i.

The approach is reminiscent of reinforcement learning (Sutton and Barto 1998; Wiering and Van Otterlo 2012): To relate our approach to this well-known learning paradigm and also to make it accessible to other researchers in this way, we discuss it with respect to the main components of reinforcement learning: states, actions, rewards, and the update scheme:

1. *States* The approach works in a continuous state space. States could be either the probability vectors with predicted (estimated) probabilities for each category or a feature representation that is derived from the probability vectors, i.e., a representation on the basis of estimated probabilities that aims to make learning easier.
2. *Actions* An action could be (a) a general type of action according to our categorization in Sect. 3.3 or (b) a specific query that is sent to the search engine. In case (a), an action is described on a rather abstract level. In other words, one would pick *the type of action*, not a specific action itself, of which there are eleven possible cases: the ones from 3.3.2 to 3.3.6 relative to either the user interest category or relative to the category with the highest prediction confidence (i.e., where the learning algorithm infers that the search engine categorizes the user) plus the random move. Variant (b) is to consider a specific query that is sent to the search engine as an action. This seems to be more complex and would require more data for training. One of the main differences between our approach and standard reinforcement learning lies in this two-step procedure: first picking the general type of action and then picking a query from the category that resulted from the first step.
3. *Rewards* In our setting, one would not receive a reward in the strict sense of the word from the environment, but observe a change in the score function that results from taking an action in a given state. The score function is a weighted

[2] This resembles the expected value for the distance between the user interest category κ_i and the assignment to an interest category by the search engine, with the difference that the categories do not exclude each other and thus the probabilities do not sum up to one.

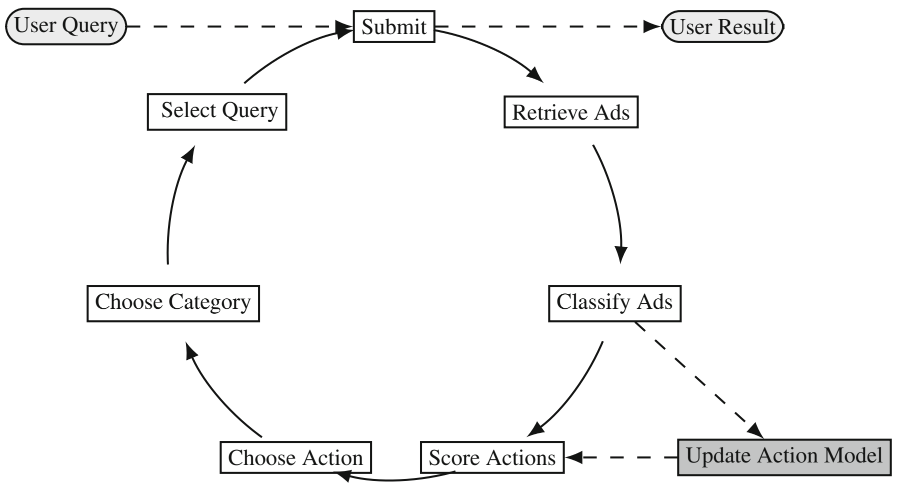

Fig. 1 Visualization of the algorithm. First, a query is sent to the search engine that represents the user's interest category. Then, the ads are retrieved from the result page. The ads are classified into the categories, such that a probability for each category is returned. From the probabilities, a feature vector for the scoring classifiers is generated. A classifier for each possible action predicts a score. The action with the *highest* predicted score is chosen (or a randomly selected action from the ones with the highest scores, in case there are multiple *top* scoring actions). The category according to the action is selected, and a query from this category is randomly selected. This query is sent to the search engine, and the process starts over again. In our experiments, randomly, from time to time, a query from the user's interest category is selected and submitted to the search engine to emulate a normal user's behavior (*marked in light gray*). After each iteration, the probabilities from the classification of the ads are used to update the previous action model (*marked in dark gray*) with the new data

sum of the distances of all categories in the category tree from the assumed user interest category. The weights are the estimated probabilities from the pre-trained classifiers. The decision is to take either the absolute value of this score function (which we chose to do) or the difference of this score between two consecutive iterations as the reward.

4. *Update scheme* In traditional reinforcement learning, one would often adjust the old value of an evaluation function, e.g., of the Q function, by a weighted difference between the old and the new value, either by just observing a trace of states, actions and rewards (as in SARSA) or by evaluating all possible actions for the next state (as in Q learning). In the approach described above, we pragmatically add the state description together with the absolute value of the score for the target variable to an incremental learning algorithm (see Sect. 3.5). More advanced update schemes could be employed as well, however, the effectiveness of the training procedure given the limited training data has to be ensured.

Figure 1 illustrates the algorithm, and Algorithm 1 shows the pseudo-code. The loop is running indefinitely long. Real user queries are entered independently from the loop and only share the same user with the rest of the process. The result of them is not used in the training of the models. Nevertheless, they are the input that the search engine uses to identify the user's interest.

```
Input: Input query $q_{in}$, interest category $\kappa_i$, tree of categories $T$
$f \leftarrow train\_models(T)$
$j \leftarrow 0$
$q_j \leftarrow q_{in}$
/* initialize default model that predicts default score of 0.0 for each
   action                                                              */
$g_0 \leftarrow init\_default\_model()$
while $true$ do
    /* Check if true browsing ratio condition is fulfilled              */
    if $u < (c + u) \times \theta$ then
        /* Get and submit user query                                    */
        $q_u \leftarrow get\_user\_query()$
        $A \leftarrow submit(q_u)$
    else
        /* Submit & retrieve ads $A$                                    */
        $A \leftarrow submit(q_j)$
    end
    $P_j \leftarrow \{\}$
    for $\alpha_l \leftarrow A$ do
        /* Generate features                                            */
        $x_t \leftarrow bag\_of\_words(\alpha_l)$
        /* Classify ads                                                 */
        $P_j \leftarrow P_j \cup f(x_t)$
    end
    $P'_j \leftarrow average(P_j)$
    if $j > 0$ then
        /* Update model from previous round                             */
        $x \leftarrow gen\_features(P_{j-1})$
        $\sigma \leftarrow score(\kappa_i, P'_j)$
        $g_j \leftarrow update\_model(g_{j-1}, x, a_{j-1}, \sigma)$
    end
    $a_j \leftarrow \operatorname{argmax}_{a_k \in A} g_j(P'_j, a_k)$
    if $a_j$ is relative to $\kappa_i$ then
        $\kappa_j \leftarrow a_j(T, \kappa_i)$
    else
        $\kappa_{max} \leftarrow find\_max\_category(P'_j, T)$
        $\kappa_j \leftarrow a_j(T, \kappa_{max})$
    end
    $q_{j+1} \leftarrow random\_select\_query(\kappa_j)$
    $j \leftarrow j + 1$
end
```

Algorithm 1: Privacy Defense

3.1 Search engines and personalized advertisement

The presented approach addresses the personalized advertisement of search engines. Most modern search engines provide an advertisement program where companies can pay for the display of personalized ads, tailored directly for the user. These ads target the user and the profile in general, as well as the current environment, e.g., the web page the ad is displayed, or time and date. The search engines generate a profile on each user of their service. These profiles are not only connected to the user account, but also based on IPs and cookies. User profiles then help the search engine to display ads that are most likely to be clicked by the user, that is, represent the users' interest in that moment the best.

To help advertisers, search engines provide simple ways to categorize their ads. These categories represent user interests and, in the best case, only ads from categories

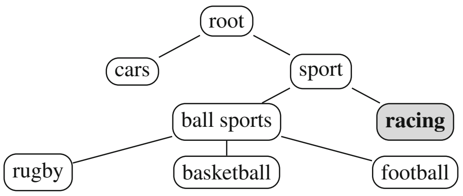

Fig. 2 Example category tree. The categories are organized in a tree hierarchy. `racing` is marked as a reference category for this example

that are relevant for the user are ultimately shown to her or him. In most cases, these categories are organized in a tree hierarchy. In this work, we used one such tree to be able to predict the categories of the ads and generate user profiles for virtual users that are interested in specific topics.

3.2 Category tree

To estimate the classification of the user on the server in terms of user profile, the feedback from the query needs to be assigned to a category. We extracted these categories from the ad program of one search engine. It provides a hierarchical tree structure for 2221 categories along with example queries for each category. A smaller example tree is given in Fig. 2. To each category of the tree, the search engine provides a set of example queries to help the advertisers assign their ads to the right category.

To classify the ads into categories, we built a static model that uses the text of the ads as input, and translated the ads into a feature vector using a bag-of-words representation. The feature vectors are then used as input to 2221 independent classifiers, one for each individual category. Hence, each trained model predicts a probability for a single category. We used support vector machines as a classifier, implemented in the WEKA workbench (Hall et al. 2009).

At first glance, it may seem counterintuitive to use a flat classification scheme instead of a hierarchical classification scheme (Ceci and Malerba 2007) for our task. However, the goal is to predict also internal nodes of the tree, not just the leaves, and only 33 out of all possible 3664 queries occur in more than one category. Therefore, it is suitable to model the classification task by a set of flat classifiers.[3]

3.3 Actions

We defined six possible actions that are used to select a category in relation to another category and the returned ads: `Random`, `Same`, `Sibling`, `Most General`, `Most Specialized of Sibling`, `Distance-Based`. All actions except the `Random` action can be calculated in two settings, in relation to the user interest cat-

[3] In the terminology of Ceci et al., we are thus using a so-called *proper training set*, not a *hierarchical training set*. Another notable difference from standard hierarchical text categorization is that our training set consists of queries, not of full documents.

egory, which is fixed, and in relation to the current category, i.e., the category that is given by the feedback (ads) of the last query. Each of these actions represent two possible actions that are treated independently. If an action returns more than one category, one action is randomly chosen from them.

In the following, we will use κ as category from node V in tree T, a_x as action function, $\kappa_i \in K$ as the user's main interest category, and κ_c as the users current (predicted) interest category, predicted from the ads. κ_{ref} can be either κ_i or κ_c, meaning this action can be defined in relation to either the current or main interest category of the user.

3.3.1 Random

The first action is the `Random` action, that simply selects a random category from the category tree. The intention is that it can be selected by and useful for the algorithm in case it is stuck in a category and cannot improve by the other actions.

$$a_{random}(T, \kappa_{ref}) = random\_select(\kappa_r \in K) \tag{2}$$

3.3.2 Same

This action simply returns the same category as the input category, either the last category chosen, or the user interest category. This can be beneficial when the user reaches an optimum, or the original interest category is chosen to walk back in the tree. In the example of Fig. 2, `racing` would be returned.

$$a_{same}(T, \kappa_{ref}) = \kappa_{ref} \tag{3}$$

3.3.3 Sibling

This action simply returns another random node that has the same parent as the reference node. The goal is to move slowly away from the reference category, making the search engine believe in a slow shift of interest. In the example of Fig. 2, `ball sports` would be chosen (another random sibling if more than one sibling could be chosen).

$$a_{sibling}(T, \kappa_{ref}) = sibling(\kappa_{ref} \in K) \tag{4}$$

3.3.4 Most general

This action chooses the most general action of the subtree of the reference category, in the example in Fig. 2, this would be `sport`. This action should confuse and mislead the search engine by pretending to be interested in a category that is similar to the reference category, as the reference category is included in the returned category, yet covers a much broader area of interest. It should be noted that the tree usually does not have a real root: there is no category on that level. That is, only categories from the second tree level are considered here.

$$a_{general}(T, \kappa_{ref}) = max\_parent(\kappa_{ref} \in K) \tag{5}$$

3.3.5 *Most specialized of sibling*

This action chooses a random sibling and follows a random path to the most specialized category. In the example in Fig. 2, one of `rugby`, `basketball`, or `football` would be randomly chosen. By this action, the user profile should be lead far away, yet still be similar due a close common ancestor. Hence, it should appear as a real shift and specialization of interest and not a random query from another interest category.

$$a_{specialized}(T, \kappa_{ref}) = lowest\_child(all\_siblings(\kappa_{ref} \in K)) \tag{6}$$

3.3.6 *Distance-based actions*

The distance-based actions choose a category based on the distance to the reference category. The result category is given by

$$a_{dist}(T, \kappa_{ref}) = \kappa_r : \forall \kappa_t \in K, d(\kappa_r, \kappa_{ref}) \geq d(\kappa_t, \kappa_{ref}). \tag{7}$$

This category is the category the furthest away from the reference category with respect to a distance measure. We implemented two distance measures, the Jaccard distance, and a distance based on normalized mutual information. This aims for a completely different category, trying to move the user far away from the reference category and making the search engine believe the interest is least similar to the real interest.

Maximum Jaccard distance The Jaccard index measures the intersection divided by the union of two sets:

$$d_J(\kappa_a, \kappa_b) = 1 - \frac{|\kappa_a \cap \kappa_b|}{|\kappa_a \cup \kappa_b|} \tag{8}$$

In our case, the union and intersection is based on the words of the search queries defined in the categories. That is, given a category, the category with the highest dissimilarity according to the Jaccard distance using the words in the two categories is chosen. This differs from most other actions, as the distance is solely based on the queries of the categories, not their position in the tree. Nevertheless, the position in the tree reflects the relationship among the categories, and therefore is in a way also related to the queries of the category. This action chooses a category that is very dissimilar in terms of words in the search queries, so it should return a category that is far away in the tree and very different from the reference category.

Minimum normalized mutual information Mutual information gives an information-theoretic measure of the dependence between two variables, or in our case, categories.

It is defined as

$$I(\kappa_a, \kappa_b) = \sum_{a \in \kappa_a} \sum_{b \in \kappa_b} p(a, b) \log \left(\frac{p(a, b)}{p(a) \cdot p(b)} \right), \tag{9}$$

where $p(a, b)$ is the probability of $(a, b) \in \kappa_a \times \kappa_b$ and $p(a)$ and $p(b)$ are the corresponding marginal probabilities. Similar to the Jaccard index, mutual information uses the queries in the categories. As mutual information does not necessary lie between 0 and 1, we used the normalized mutual information (NMI) as a measure:

$$NMI(\kappa_a, \kappa_b) = \frac{I(\kappa_a, \kappa_b) \cdot 2}{H(\kappa_a) + H(\kappa_b)}, \tag{10}$$

where H is the entropy. The corresponding distance function then is:

$$d_{NMI}(\kappa_a, \kappa_b) = 1 - NMI(\kappa_a, \kappa_b) \tag{11}$$

This, similar to the Jaccard distance, should return a category very different based on the queries in the categories, and hence, far away in the tree.

3.4 Retrieve ads

In the next step, the ads are extracted from the resulting web page. Most modern search engines return personalized ads based on a search query, the location of the user, the general interest of the user, and the current environment. These ads are tailored towards the user and the query that was just submitted. Additionally, the ads are clearly marked on the web page, visually, as well as in the HTML source code. Therefore, the ads can be easily extracted.

The ads themselves usually consist of a text and a URL. The text is targeted towards the user's interest, so that the user is attracted to click the ad. This makes it easy to identify the topic of the ad. Additionally, the ads tend to be written not in full texts, but due to the limited space, in a more compact, bullet point type, fashion.

We only consider the first page of the search results, which typically contains multiple ads. Each add is given to the category classifier and predicted independent of each other. Next, the predictions of all ads are averaged. This is done to aggregate the varying number of ads that are returned into a feature.

In the case when no ads are returned, the step is not used in the algorithm, as there is no feedback to be used. Additionally, one cannot say that this is a good thing or a bad thing. It could be that the user is between two categories such that no ad can be displayed that fits both interests, or that simply there are no ads available at the moment that fit to the user profile. In any case, the algorithm cannot use any information from this type of no-ads event.

3.5 Action scoring and selection

Several actions are possible to choose a category for the next query, all defined in Sect. 3.3. To score each action, we trained online linear regression models using stochastic gradient descent based on the classification of the ads from the previous query (g_i in Algorithm 1). The models were implemented using WEKA. In the training phase, a score is calculated from these classifications that is used as target variable.

The models are updated whenever the corresponding action is used, i.e., new training data are available. The score is given in 1 and based on P the vector of probabilities for the categories based on the prediction of the models on the ads, and the category the user is interested in κ_i. Each category $\kappa_j \in K$ has a corresponding probability $p_j \in P$. Finally, $d_T(\kappa_i, \kappa_j)$ is the normalized distance between categories κ_i and κ_j in a T (distance related to the tree structure, i.e., each edge between the categories or nodes adds with a value of one to the distance).

The features are generated from the vector of probabilities as follows: First, the score from this step is calculated and included as a feature. Second, the distance in the tree to the category with the highest probability is added. Third, the probability of the category that is the furthest away in the tree from the user interest category. Fourth, the distance between the root and the lowest common ancestor of the user interest category and the category with the highest probability is added. And finally, a set of features is added which gives the average probability over the categories at a certain tree distance to the user interest category. That is, one feature that averages all probabilities of categories one edge away from the user interest, the next averages over all categories two edges away, and so forth. Table 1 gives a summary of the features used. We calculate this set of features for the previous five queries and use them in combination.

The classifier is updated each time the corresponding action is selected. After the action is chosen, a query from the corresponding category is used to query the search engine. After the resulting ads are evaluated using the pre-trained models, the score is calculated and the predicted probabilities together with the score as the target value is used for training.

Table 1 Summary of the features used in training

Feature	Description
Previous score σ	The score σ of the previous query
Distance to highest probability	Distance in tree to the category with the highest probability, each edge between the categories adds 1
Furthest probability	Probability of the furthest category from the last query (furthest in terms of distance in the tree)
Depth of LCA	Distance of the lowest common ancestor to the root
Average probability per distance	One feature per distance each feature with the average probability over all categories with that distance

Note that we used these generated features from the previous five steps, i.e., the algorithm calculates five times these features and combines them into a training or testing instance

The prediction of the score is straightforward: The trained models are simply applied to the predicted probabilities from the retrieved ads. The model then returns the predicted score for the corresponding action. If two or more predicted scores are the same, one action is selected randomly. However, in the experiments that never happened due to the predicted scores having a large number of digits.

The final step in the algorithm is the selection of the query. The action that was selected in the previous step is carried out and $a(T, \kappa)$ returns a new category. In this category, multiple queries are available. The algorithm randomly picks a query and submits it to the search engine.

4 Experiments

To evaluate our approach, we set up a system of multiple components that emulate a user submitting queries to a search engine and retrieve results in terms of ads from this search engine.[4] There are several technical requirements for the evaluation that we had to address:

- The method needs a way to create distinguishable user profiles such that the search engine recognizes the users.
- The evaluation cannot use any real ground truth, as this is only available to the search engine itself.

After a query is submitted, the system extracts the ads from the web page by parsing the HTML (the ads are clearly marked by a distinguishable environment). The algorithm described in Sect. 3 is then used to generate new queries and update the models.

To address the first problem, we created users with unique cookies from the search engine and unique browser fingerprints, so that the search engine easily can recognize the user. We did this for all users, giving each user a unique profile.

The second problem is harder because it affects the evaluation process. We do not know what the search engine "thinks" the user is interested in or what profile the user has. We can just estimate it from the returned ads and predict classes using the model trained in advance from the category tree. However, this should give an idea of the search engine's categorization. From this, we can calculate the score that is also used in training (see Eq. 1). Another possibility is to check the probability of the user main interest category, to see if the ads are targeted towards that at all. To evaluate these measures, we created two other sets of users that:

1. Just queried the search engine with random queries, and in 10% of the cases used a query from the interest category.
2. Just queried the search engine with queries from the most distant category in relation to their interest categories, and in 10% of the cases used a query from the interest category.

The remaining users behaved similarly: In 10% of the cases, they submitted queries from their interest category, in the remaining cases, they submitted queries according

[4] The implementation is available upon request.

to the introduced approach. The case where the user submits queries from the interest category mixed with a fixed other interest category has been proposed in related work (Rebollo-Monedero et al. 2012). Note that in our experiments, we set the true browsing ratio θ to 10%, as this is a good simulation for a real-world application: For every query of a user, nine generated queries are submitted. This is still at a level that it is not obvious for the search engine that most queries are automatically generated (the search engine could block the user if there were too many queries submitted by the same user).

While there exist a few search engine data sets that are available for research, e.g., the Yandex data set at Kaggle, this cannot be used in our experiments. We target a complex black box, not knowing what algorithms exactly the search engine uses to identify users, hence we cannot recreate the search engine's algorithms. In this way, this work differs from similar approaches (Beato et al. 2013; Nikiforakis et al. 2015) that targeted a simple algorithm or technique.

In related work (Rebollo-Monedero et al. 2012; Sánchez et al. 2013), entropy was used to calculate a measure of query obfuscation:

$$H = \sum_{i=1}^{|Q'|} p(\rho_i) \cdot logp(\rho_i) \tag{12}$$

With the full set of distinct queries Q' of a user and single queries $\rho_i \in Q'$. It calculates the entropy over the queries for every user, $p(\rho_i)$ is estimated by the fraction of cases in which this query occurs in the queries $q_j \in Q, j \in 1, \ldots, i$ of that user. A high entropy represents a wide range of queries, and therefore it might be harder for the search engine to identify the users interest as there is more noise in the data. However, the data mining algorithms used by the search engine are tailored to find the signal in noisy data. So entropy can also only give indications of the performance of a method and not reliably completely evaluate it on its own.

4.1 User categories

We chose 20 categories for the users and let each category be the main interest category for 12 users. The interest categories are listed in Table 2. We selected categories that should have a strong attraction for the advertisement industry, e.g., the `family` category that summarizes queries related to any family topic, such as baby and children's products. Due to this, there should be numerous advertisers in these categories, which leads to a higher number of ads returned, and therefore more data for the experiments.

5 Results

5.1 Performance of the Ad classifiers

As the whole evaluation and the model relies on a good pre-trained classifier for the ads, we evaluated the classifier in two scenarios: First, we used standard 33%

Table 2 The list of selected categories used in the experiment

Categories	
Antiques and collectibles	Bicycles and accessories
Car video	Computer components
Cosmetic procedures	Dating and personals
Desktop computers	Drugs and medications
Erectile dysfunction	Family
Game systems and consoles	Laptops and notebooks
Make up and cosmetics	Motorcycles
Real estate listings	Sexual enhancement
Timeshares and vacation properties	Toys
Vitamins and supplements	Weight loss

Each category was used by 12 users, plus 12 users that submit random queries mixed with queries from the interest category, and another 12 users that submit queries from their interest categories combined with queries from the category that is the furthest away in the tree

Table 3 Performance of the static pre-trained ad classifiers using a 33% holdout set

Class	AUROC	Precision	Recall	F-measure	AUPRC
1	0.9844	0.8338	0.7493	0.7816	0.8033
0	0.9844	0.9999	0.9999	0.9999	0.9999

Due to the strongly imbalanced class distribution, the performance on the negative class is far better than on the positive class. Yet the performance regarding the positive class is still on a good level and usable for our task

holdout evaluation on the training set consisting of the categories and the corresponding queries. Second, we applied the trained models on randomly selected ads from our experiments and annotated them manually with the correct categories. In the first case, we applied the models that were trained on queries to a test set of the queries. In the second case, we applied the models to actual ads, the real-world scenario used in the main method. In both cases, we used multi-label classification (one label for each category) and micro-averaged the predictions over all target values to calculate the measures area under ROC curve, Recall, Precision, F-Measure, and area under precision recall curve.

Table 3 gives the results of the 33% holdout evaluation. As the training set is highly imbalanced (all possible queries of all categories are used, yet only around 594 on average are positive, almost all examples are negative, the size of the data set is 1,118,134 instances). Hence, the negative class is easy to predict, and the performance regarding it is rather high. Nevertheless, even the evaluation regarding the positive class is still at a good level. The area under the precision recall curve is rather high, the F-Measure of 0.78 also shows that the classification works to some satisfactory extent.

We additionally evaluated the classifiers using a manual labeling of the ads. To that purpose, we randomly sampled 100 cases and compared the predictions of the classifiers to manually assigned categories. The classifiers achieve an area under ROC curve of 0.844 and an area under precision recall plot of 0.739. This shows that the prediction of the categories from the labels works rather well.

The evaluation shows that models perform quite well on both tasks. This is most likely due to the similarity of the ads and the search queries. That is, words are chosen that either attract the attention of the user (ads), or are most likely to find web pages that belong to that category (queries). In both cases, the words need to be very representative of that category. This simplifies the classification and leads to respectable classifier performance.

This shows that, in a real-world setting, it is possible to estimate the user profile based on the displayed ads. In general, the ads use strong keywords to attract the attention of the user, and beyond the annotated data used in this setting, unsupervised text mining could be used to predict and identify categories of ads that directly reflect the user profile at the search engine.

5.2 Performance of the method

As explained above, evaluating the approach is quite hard as we do not have ground truth. We can just estimate the performance and see if the system behaves the way we expect it to behave. Therefore, we cannot say for sure whether the approach was successful; yet the results indicate that it has shown some desired effect.

Figures 3 and 4 gives an overview on the results.[5] Note that due to space limitations, we only give selected results, all results are given in the supplementary material. We aggregated the data points for score and probability over 50 queries to reduce the variance in the plots. Furthermore, we only included data points where at least half of the users submitted queries and got ads from the search engine. This leads to different lengths of the lines as some scenarios lead to fewer displayed ads. It should be noted that the values on all users were averaged and normalized over the user categories before averaging them into the plots.

In general, the score (see Eq. 1) of the users that only submit random queries is more or less stable at a certain level. The same is true for the probability for the user interest category. On the other hand, with trained models, the score and probability changes in various ways. In most cases, the score increases while the probability decreases for queries not in the user interest category. At first, the score and probability strongly improve, then become more stable, but, in most cases, still improve slowly. This can be attributed to the online models that, in the beginning, do not have enough training data and the best action is not necessarily chosen. Over time, the models improve and better actions are chosen, and it appears that the users slowly move away from their user interest category, which was one of the main goals of the algorithm. It should be noted that the models could be pre-trained with existing data and have a good performance from the beginning. However, in the plots we show the performance from a cold start, i.e., all the results we got in the experiments.

Users using only the most distant category for generating queries can achieve the highest scores. However, the probabilities for the interest categories show the flaw of this approach: They are in general rather high and tend to even increase over time. This means that the search engine assigns them more and more to their interest category,

[5] Detailed results and statistics on the results are given in the supplementary material.

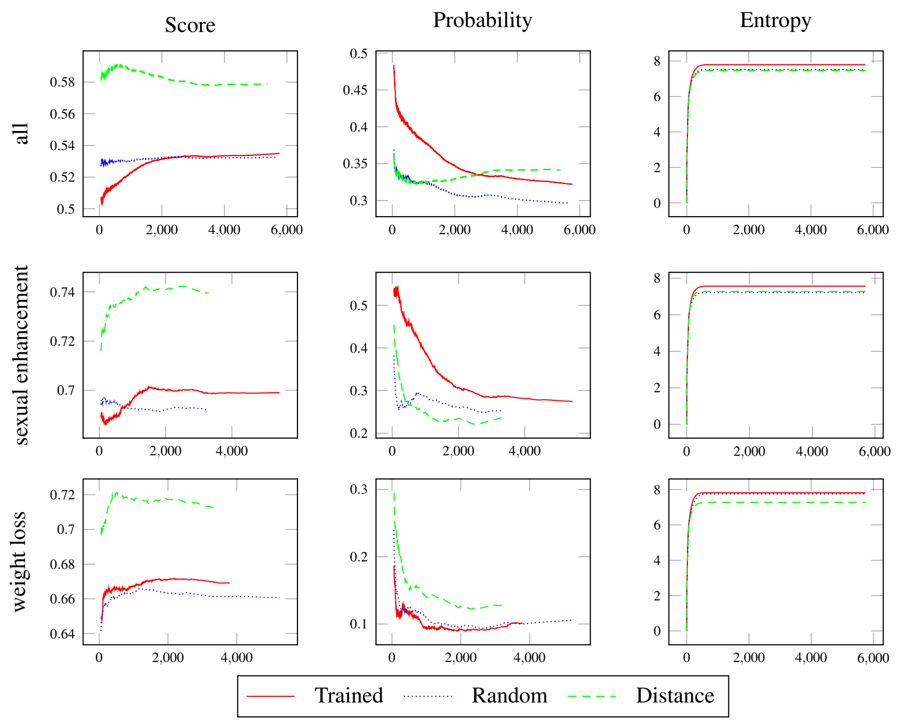

Fig. 3 Results of the experiments regarding queries **not** in the user interest category. The score (see Eq. 1), entropy of the queries, or predicted probability (which one is used is given above the *top row*) for the user interest category depending on the *y*-axis (the category is given *left* of each *row*), compared to the queries on the *x*-axis. The entropy is calculated over all the queries used so far. Each row summarizes the users of a category (or all users in the first row). Trained refers to queries generated by the proposed algorithm, Random refers to queries by users that exclusively submit random queries and queries from their interest category. Distance are queries from users that submit queries from their interest categories and queries from the most distant category. The plots give the results of queries **not** in the user interest category only

while, at the same time, assigning them to the second category they use. This shows that the search engine actually uses a more sophisticated model, as discussed in Sect. 2. Nevertheless, this leads to the search engine being able to identify the main interest of the user regardless of other queries. The trained model is able to reduce the probability over time, as the only approach that is presented here. This means that the search engine believes less and less that the user is actually interested in his or her own category. This becomes especially visible in the case of queries from the user interest category. This shows that this approach, variants of which were previously proposed, can be improved by using the information of the ads when obfuscating the user profile.

Another perspective on the results regarding the probability of the user interest category is that a high probability indicates that the user is recognized by the search engine to be interested in that category. The strong decline of the generated queries from a high value of above 0.3 for all users shows that the user is indeed in the beginning identified as having an interest in the interest category. This assignment strongly degrades for the generated queries, however, it increases for the case of

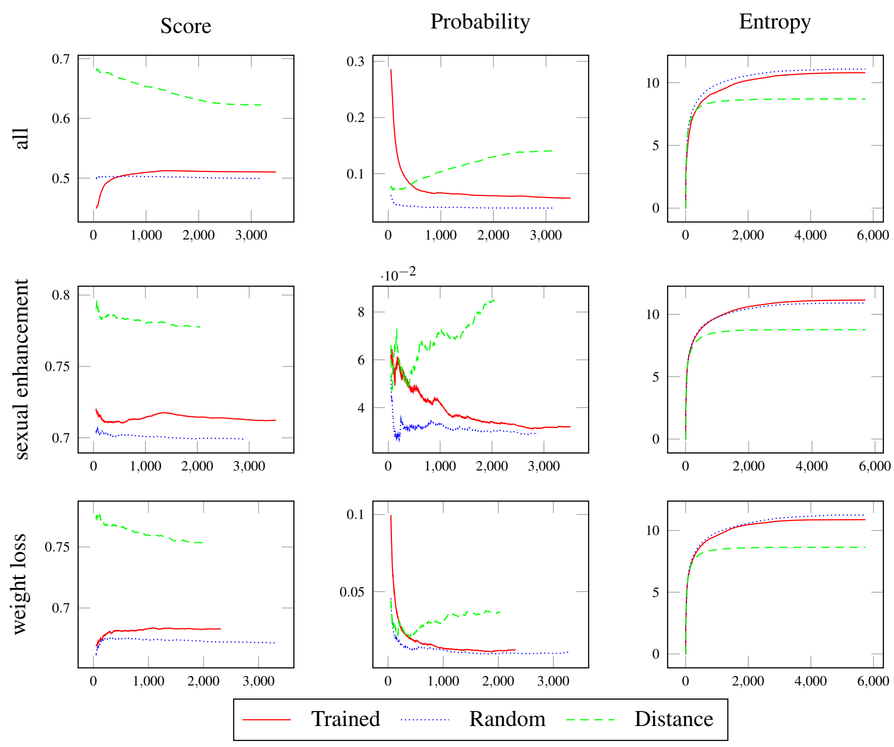

Fig. 4 Results of the experiments regarding queries in the user interest category. The score (see Eq. 1), entropy of the queries, or predicted probability (which one is used is given above the *top row*) for the user interest category depending on the y-axis (the category is given *left* of each *row*), compared to the queries on the x-axis. The entropy is calculated over all the queries used so far. Each row summarizes the users of a category (or all users in the first row). Trained refers to queries generated by the proposed algorithm, Random refers to queries by users that exclusively submit random queries and queries from their interest category. Distance are queries from users that submit queries from their interest categories and queries from the most distant category. Due to the different ratio of queries resulting in ads, the number of data points differs for each plot. Note also that the scales differ depending on the category. We decided to use different scales to improve the visibility, as the score and probability strongly depend on the category and differ from category to category. The only important factor is the relation between different approaches in the same category. Note that full results are given in the supplement

using queries from the most distant category. This supports the assumption that the search engine assigns users to categories based on their search queries, otherwise, this probability would not rise.[6] For the trained models, it shows that the actions slowly move the user to other categories, staying close to the original category, e.g., by choosing mainly actions like `sibling`. However, also this is not the case. The dominant actions in the beginning are distance-based actions, hence categories are chosen that are very different from the interest category.

[6] This could only be the case when the `same` action with regard to the user interest category would be chosen, which is not the case. This action almost never gets chosen, as it simply never is evaluated by high scores (details not shown here).

Furthermore, the result plots show that the strategy of showing ads differs depending on the category of the ads. Ads that are more controversial, as from the `sexual enhancement` or `erectile dysfunction` (see supplement) category seem to be displayed only in cases when the search engine is sure about the user. It is easier for the trained model to not get ads from these categories even if the main interest lies in it. The decrease is much stronger than in the cases of more decent categories like `weight loss` or `toys` (see supplement). Additionally, the method succeeds in achieving a lower probability and higher score compared to random for generated queries. In the case of user interest queries, it seems that the user categorization is still adapting and changing, potentially at one point beating the other methods. At the same time, in the case of `sexual enhancement`, both the random and most distant method seem to stabilize while the trained queries seem to still improve.

In general, the trend appears to be that the trained models produce queries that improve the score and probability, the random queries stay at a stable level, and the most distant queries improve the score but the probability gets higher over time. The trained models find a balance between low probability and high score, hence achieving a higher level on privacy than the compared methods that only optimize one measure at a time. Just achieving a high score can also mean that the search engine assigns the user to one further category that is far away from the user interest category, but also recognizes the user's interest in his or her defined category. Similarly, just a high probability would only mean that the specific user category is not recognized very well, however, closely related categories would still be assigned to the user. Both cases mean that the overall goal of obfuscating the user's profile would not be achieved. Therefore the trained models are superior to the compared methods and, most importantly, it is possible to evoke an intended reaction from the search engine.

In related work (Rebollo-Monedero et al. 2012; Sánchez et al. 2013) the entropy of the submitted queries was used as means of profile obfuscation. We give the entropy plotted over the number of queries. While for obvious reasons, the entropy on the user interest category is the same for all approaches, there is a clear difference on the entropy in queries not from the user interest category. As expected, the approach using only two categories has the lowest entropy, as only queries of these two categories are considered. The random and the trained approach produce, on the other hand, a higher entropy, which is very similar for both cases. This makes sense, as in both cases, various categories are used for query selection and this increases the entropy compared to the other approach.

The results show that there actually appears to be a relation between the entropy of the queries and the proposed score, as well as the calculated probability for the interest category. A lower entropy in many cases indicates a higher probability and lower score when comparing the random and trained approach. This backs up our assumption that the search engine actually gives an indication of what it knows about the user in the advertisements.

Another aspect of the model is which actions are actually chosen. If only a single action is chosen, the trained models would not be necessary and only this action could be used. However, Table 4 shows that this is not the case. The table gives the counts of each action that was used in the experiments after a certain time. It should be noted that this also includes actions that resulted in an empty set of ads and were not used

Table 4 The counts of each action, i.e., how often each action is chosen by the algorithm

Action	Counts $\kappa_x = \kappa_c$	Counts $\kappa_x = \kappa_i$	Counts total
$a_{d_J}(T, \kappa_x)$	2939	2711	5650
$a_{d_M}(T, \kappa_x)$	4634	76,192	80,826
$a_{general}(T, \kappa_x)$	336	97	433
$a_{specialized}(T, \kappa_x)$	26,262	1954	28,216
$a_{sibling}(T, \kappa_x)$	15,435	245	15,680
$a_{same}(T, \kappa_x)$	37,426	183	37,609
$a_{random}(T, \kappa_x)$	–	–	154,472

Even though the score relies solely on the distance in the tree, an action based on mutual information has the second highest counts

in the further evaluation. Notably, even though the score solely relies on the distance in the category tree, the action using the mutual information of the category queries is the second most used action after random. This could indicate that the search engine actually uses mutual information or something similar in their user categorization. Furthermore, random queries are apparently helpful for the algorithm, and using a sibling or the most general parent in the tree improves the performance.

6 Related work

Similar to this work, Rebollo-Monedero et al. (2012) addressed query profile obfuscation in a search engine from a user perspective. While they adapted a more sophisticated user model with multiple interest categories, the exchange among the users is rather simple and more of a theoretical nature. Their approach is similar to our method that uses only one category to use queries from ($mostDistant$). Similarly to Rebollo-Monedero et al. (2012), Sánchez et al. (2013) and Viejo and Sánchez (2014) focus on the entropy of submitted queries. Both works exploit social networks to extend the set of queries of a user.

Furthermore, Howe and Nissenbaum (2009) introduced Trackmenot, a browser extension aiming to confuse the search engine by automatically submitting queries. In its core, it is very similar to the Random query submission approach used in our experiments as it simply submits queries from a set of queries that is modified over time, but due to the way it is modified, the queries remain rather similar. Additionally, in newer versions, Trackmenot implements further steps like automated clicks on the results page that can be combined with the proposed approach in future work. None of these papers use feedback from a search engine to adapt the submitted queries.

Gervais et al. (2014) introduced scores that quantify the privacy of obfuscation techniques. However, we cannot apply them to the evaluation as it relies on a linkage attack on the side of the search engine, information that we do not have. Nevertheless, they relate to the score σ that we introduced.

The closest related work is adversarial learning, which tries to defend machine learning algorithms against an adversarial opponent (Huang et al. 2011). Research in that field also sometimes takes the role of the adversarial opponent. This work introduces the counterpart of adversarial learning. We aim to provide technologies to actively confuse data mining and machine learning algorithms. This could be understood as some form of adversarial learning, yet the goal is the opposite.

In the context of adversarial learning, Barreno et al. (2006, 2010) and Huang et al. (2011) introduced a framework and taxonomy to categorize attacks on machine learning algorithms. In their attack model, the presented approach could be categorized as `exploratory`, as we do not influence the training process in the search engine directly, `indiscriminate` as the overall goal is not targeted but to just get anywhere not related, and `integrity`, as confusing queries should be classified as normal. The authors propose defense mechanisms against these attacks. In these categories, they suggest information hiding and regularization as counter measures. Both are not easy to implement against the presented approach. Due to the online learners, the actions can adapt to avoid being identified and choose different actions until the score rises again. Hiding information seems not to be an option, as it is hard to identify the generated queries, and the ads cannot be hidden to all users.

Huang et al. (2011) used several case studies to evaluate the threat of adversarial attackers. In contrast to the presented work, they use scenarios where the learner is known and can be easily influenced by submitting data to it. Due to the size of the data available to modern search engines, there is no chance of manipulating the models in our approach, hence we solely rely on false predictions of the search engines models by generating false data using data mining techniques.

Multiple other publications address the problem of malicious noise in the data (Klivans et al. 2009; Biggio et al. 2012; Frénay and Verleysen 2014). Nevertheless, they do not target an adapting attacker towards an unknown system, as in the case of this work.

Lowd and Meek (2005) might be the closest related to this work. They propose a framework for a problem class into which this work falls as well. However, their chosen problem, spam filters, is much easier to attack as they use local models and can train and evaluate frequently. Although they again target the opposite direction, namely defending algorithms against adversarial attacks, our work falls into their ACRE learning framework, but uses a much more adaptable approach, as we use direct feedback from the search engine while Lowd and Meek did not use any direct feedback from the spam filters.

Beato et al. (2013) and Nikiforakis et al. (2015) targeted similar goals in other domains, however, only focused on well-researched goals with a known algorithm. This work targets an unknown algorithm where the feedback is hard to get and understand.

A similar goal to this work, yet a completely different approach, is followed in the field of privacy-preserving data mining (Agrawal and Srikant 2000; Skarkala et al. 2012; Verykios et al. 2004; Kargupta et al. 2003; Pedreschi et al. 2008). However, as discussed above, this requires the user to trust the companies to transform the data. In contrast to this, we want to give the user the power to defend his or her privacy using data mining tools.

It should be noted that there exist several privacy protecting search engines. Ixquick (see https://ixquick.com/) for example provides a service called Startpage (https://

startpage.com) that acts as a proxy, Duckduckgo (https://duckduckgo.com) works in a similar way. This efficiently hides the user's identity from other engines, as the user's queries are mixed with queries from other users. Yet, again the user has to trust this company with the user's data. Another approach is the use of anonymization techniques like TOR, that hide the IPs and make it possible to obfuscate the complete traffic. However, service providers could still identify users using cookies or browser fingerprints.

7 Conclusion

In this paper, we presented a new approach to give a user the possibility to defend her or his privacy on the Internet. We introduced a new method and evaluated the method in a challenging real-world environment. While the results indicate that the approach is feasible, we cannot be sure about this, as we do not have a ground truth to evaluate against. However, the work should be understood as suggesting a new class of methods that help users directly in contrast to previous work on privacy-preserving data mining, which works on the service providers' side. The presented algorithm can be adapted to many real-world applications targeting a similar problem.

While this work uses search engines as a case study, these are not the main target for the presented approach. Major search engines are honest and clear about user profile data usage and provide a free service that is paid in a way by the users' data. Nevertheless, this scenario provided a good environment to explore the possibilities of the approach, as search engines have excellent technologies to categorize a user and provide feedback via personalized ads.

Furthermore, we used a simplified target model for this work. In general, users have interest in more than one category, or might not want certain categories to be misplaced. Also, a search engine uses a complex modeling for the users and the displayed ads depend on bidding of interested parties. Also other factors, such as time and date of the queries play a role. However, the goal of this work was to evaluate if the presented approach can trigger some kind of reaction by the search engine. The accumulated responses in terms of ads show that we can indeed modify the response of the search engine even with our simplified target model. Future work can address more sophisticated scenarios, where the user's interest is in multiple categories, more factors play a role when using the response, or some categories are unwanted for the user. Additionally, the user might have some interest in keeping the personalized search results while, at the same time, reducing the personalized ads: A first step might be adapting the scoring function to target this.

The results in this paper are promising, yet a vast amount of application areas and improvements to this approach are possible. The approach at the moment strongly relies on the feedback of the search engine. An extension would be to make it more independent from this feedback and use other input to the scoring function, which would enable it to work under more and more general conditions. Also, data may come in many different forms: While we were discussing the approach in terms of plain text and search queries, a lot of communication relies on multimedia data, or simply connections in social networks. All of these could be used in scenarios of future work.

Acknowledgement The authors thank Nicolas Krauter for the help on the initial implementation.

References

Agrawal R, Srikant R (2000) Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data. ACM, New York, pp 439–450

Aldeen YAAS, Salleh M, Razzaque MA (2015) A comprehensive review on privacy preserving data mining. SpringerPlus 4(1):694

Barreno M, Nelson B, Joseph AD, Tygar J (2010) The security of machine learning. Mach Learn 81(2):121–148

Barreno M, Nelson B, Sears R, Joseph AD, Tygar JD (2006) Can machine learning be secure? In: Proceedings of the 2006 ACM symposium on information, computer and communications security. ACM, New York, pp 16–25

Beato F, Conti M, Preneel B (2013) Friend in the middle (fim): tackling de-anonymization in social networks. In: IEEE international conference on pervasive computing and communications workshops (PERCOM Workshops), pp 279–284

Biggio B, Nelson B, Laskov P (2012) Poisoning attacks against support vector machines. In: Proceedings of the 29th international conference on machine learning (ICML-12), pp 1807–1814

Bilenko M, Richardson M (2011) Predictive client-side profiles for personalized advertising. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, pp 413–421

Ceci M, Malerba D (2007) Classifying web documents in a hierarchy of categories: a comprehensive study. J Intell Inf Syst 28(1):37–78

Eckersley P (2010) Privacy enhancing technologies: proceedings 10th international symposium, pets 2010, Berlin, Germany, July 21–23. In: Atallah MJ, Hopper NJ (eds) Privacy enhancing technologies, chapter How Unique Is Your Web Browser? Springer, Berlin, pp 1–18

Frénay B, Verleysen M (2014) Classification in the presence of label noise: a survey. IEEE Trans Neural Netw Learn Syst 25(5):845–869

Gervais A, Shokri R, Singla A, Capkun S, Lenders V (2014) Quantifying web-search privacy. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, CCS '14. ACM, New York, pp 966–977

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor Newslett 11(1):10–18

Howe DC, Nissenbaum H (2009) Trackmenot: resisting surveillance in web search. In: Kerr I, Steeves V, Lucock C (eds) Lessons from the identity trail: anonymity, privacy, and identity in a networked society, vol 23. Oxford University, Oxford, pp 417–436

Huang L, Joseph AD, Nelson B, Rubinstein BI, Tygar J (2011) Adversarial machine learning. In: Proceedings of the 4th ACM workshop on security and artificial intelligence. ACM, New York, pp 43–58

Kargupta H, Datta S, Wang Q, Sivakumar K (2003) On the privacy preserving properties of random data perturbation techniques. In: Third IEEE international conference on data mining, pp 99–106

Klivans AR, Long PM, Servedio RA (2009) Learning halfspaces with malicious noise. J Mach Learn Res 10:2715–2740

Lowd D, Meek C (2005) Adversarial learning. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, New York, pp 641–647

Nikiforakis N, Joosen W, Livshits B (2015) Privaricator: Deceiving fingerprinters with little white lies. In: Proceedings of the 24th international conference on world wide web. International world wide web conferences steering committee, pp 820–830

Nikiforakis N, Kapravelos A, Joosen W, Kruegel C, Piessens F, Vigna G (2013) Cookieless monster: exploring the ecosystem of web-based device fingerprinting. In: IEEE symposium on security and privacy (SP), pp 541–555

Pedreschi D, Bonchi F, Turini F, Verykios VS, Atzori M, Malin B, Moelans B, Saygin Y (2008) Privacy protection: regulations and technologies, opportunities and threats. In: Giannotti F, Pedreschi D (eds) Mobility, data mining and privacy: geographic knowledge discovery. Springer, Berlin, pp 101–119

Purcell K, Brenner J, Rainie L (2012) Search engine use 2012. Technical report, Pew Internet and American Life Project Washington

Rebollo-Monedero D, Forné J, Domingo-Ferrer J (2012) Query profile obfuscation by means of optimal query exchange between users. IEEE Trans Dependable Secure Comput 9(5):641–654

Sánchez D, Castellà-Roca J, Viejo A (2013) Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines. Inf Sci 218:17–30

Skarkala ME, Maragoudakis M, Gritzalis S, Mitrou L, Toivonen H, Moen P (2012) Privacy preservation by k-anonymization of weighted social networks. In: Proceedings of the 2012 international conference on advances in social networks analysis and mining (ASONAM 2012), ASONAM '12. IEEE Computer Society, Washington, DC, pp 423–428

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction, vol 1. MIT Press, Cambridge

Verykios VS, Bertino E, Fovino IN, Provenza LP, Saygin Y, Theodoridis Y (2004) State-of-the-art in privacy preserving data mining. ACM Sigmod Record 33(1):50–57

Viejo A, Sánchez D (2014) Profiling social networks to provide useful and privacy-preserving web search. J Assoc Inf Sci Technol 65(12):2444–2458

Wiering M, Van Otterlo M (2012) Reinforcement learning. In: Adaptation, learning, and optimization, vol 12. Springer Berlin Heidelberg

Xu L, Jiang C, Wang J, Yuan J, Ren Y (2014) Information security in big data: privacy and data mining. IEEE Access 2:1149–1176