A Self-Adaptive Classification for the Dissociating Privacy Agent

Marc Juárez
IIIA - Institut d'Investigació en Intel·ligència Artificial
CSIC - Consejo Superior de Investigaciones Científicas
Campus de la UAB, 08193 Bellaterra, Catalonia, Spain.
Email: mjuarez@iiia.csic.es

Vicenç Torra
IIIA - Institut d'Investigació en Intel·ligència Artificial
CSIC - Consejo Superior de Investigaciones Científicas
Campus de la UAB, 08193 Bellaterra, Catalonia, Spain.
Email: vtorra@iiia.csic.es

***Abstract*—This paper describes an extension of the Dissociating Privacy Agent (DisPA), which is a Privacy Enhancement Technology (PET) for web search. It is implemented as an add-on for Firefox that acts like a proxy between the user and the search engine. A fundamental part of DisPA is the classification of queries into a set of categories. Particularly, the taxonomy of the Open Directory Project (ODP) is used for this purpose. In this paper we briefly recall the internal operations of the agent, discuss the drawbacks of the current model and propose an improvement to overcome them.**

I. Introduction

Web search service providers, commonly known as search engines, are the main actors on the Internet today, as they are the fastest and most effective way of finding information. They log data about users and their searches at the server side for various purposes. On the one hand, they want to exploit business opportunities by means of Marketing Research and Targeted Advertising [1]. On the other hand, due to the growth of the Web and the increasing number of results that may be retrieved for one search, it has emerged the need of improving ranking algorithms towards a more efficient search [2].

The most common way to cope with this problem is to analyze collected data in server logs in order to build a profile of the user. By applying data mining techniques in the server logs, search providers extract traits of the user such as demographic aspects (e.g. age, gender or nationality) or main areas of interest, that are modelled as categories like "Music" or "Football". Afterwards, the ranking algorithms rearrange the results list to deliver first those that are more useful according to these preferences [3].

Nevertheless, the logging of all these data for profiling entails a serious threat for the user's privacy, as it has been evidenced several times in the past. One relevant case in this line was the AOL search data leak in 2006 [4]. The research department of AOL, in a move directed to help the research community, released a dataset containing twenty million web queries for over 650,000 users over a 3-month period [5]. Although AOL themselves intended to anonymize the dataset by removing some compromising fields (e.g. user names), two journalists of the New York Times managed to discover one of the identities of the users using exclusively the terms in their queries [6]. This was very remarkable because it proved that queries by themselves may be used to uniquely identify an individual or, at least, reduce the search space considerably.

For all these reasons, the research framework where this paper is included aims to develop a Privacy Enhancement Technology (PET, for short) for web search. In the past decade several PETs have been developed pursuing this objective (for example, [7], [8], [9], [10], [11], [12], [13]), and the trend is shifting to social networks because privacy risks are more obvious (see for instance [14], [15], [16], [17]).

Our approach is characterized by taking into account search personalization. We assume that personalization benefits the user and consider the trade-off between the use (personalization) and the cost (privacy) of releasing data. In our approach, data refers exclusively to queries, since we assume problems associated to other data as the IP address or browser fingerprints as solved, and hence we assume that the adversary has no background knowledge to use in conjunction with the server log.

The rest of the paper is organized as follows. Section II reviews the state-of-the-art in private information retrieval oriented to Web and recalls the basic operation of the Dissociating Privacy Agent (DisPA for short) [18]. Section III describes the self-adaptive classification mechanism. We evaluate in Section IV this new approach and present the results obtained. In Section V we discuss the limitations and give some lines of future work. Finally, we offer our conclusions in Section VI.

II. Related work

The PIR problem was first introduced in 1995 by Chor, Goldreich, Kushilevitz and Sudan [19] in the theoretical setting and in 1997 by Kushilevitz and Ostrovsky [20] in the computational setting. It was stated as a user that wants to retrieve an item from a database without revealing which item he is actually retrieving. PIR protocols that tackle this problem are based in assumptions that are not feasible when it comes to retrieve information from the Web. For instance, they often assume cooperation by the provider. However, search providers are reluctant to add extra operations because it would increase the response time. Another important difficulty that makes them inconvenient is the computational complexity of these methods given the huge size of the Web.

978-1-4673-5839-2/13/$31.00 ©2013 IEEE

For this reason, the constraint of privacy is very often relaxed towards more applicable strategies. These other approaches allow providers to know the terms of the query that the user is submitting but attempt to hide which is the true web site that the user is looking for. A family of techniques grounded on this approach are named obfuscation-based [21] techniques which, at the same time, are often classified in two groups: (i) profile-based, trying to hide the real profile of the user by submitting false queries (e.g. TrackMeNot [8]) and (ii) query-based, which aims to hide the real query by generating bogus terms and adding them to the search request (e.g. GooPIR [7]).

There is another line of research that intends to hide the identity of the user (user anonymity) rather than the information that he is attempting to retrieve. Most of the strategies that we have found in the literature on user anonymity are based on mixing. For instance, the User Private Information Retrieval (UPIR) [9] is a peer-to-peer community of users that submit queries ones on behalf of others.

Besides of the particular shortcomings that each of these approaches have, there exist one drawback in common. All of them diminish the quality of server logs for profiling. On the one hand, obfuscation-based techniques introduce false information about the preferences of the user, on the other hand, mixing techniques submit queries of different users in the same log.

In the following subsection we are going to recall the basic operation of DisPA, our proposal for a PET in web search that preserves personalization.

A. The Dissociating Privacy Agent

First of all, we are going to describe how the system that provides the service to the user is modelled. The user sends a query using an HTTP connection to the search engine using a cookie that contains a user unique identifier. We make the assumption that search engines only use cookies to identify users. This might be a strong assumption to hold but the last version of the privacy policy and a recent study of the log retention policies support it [22], [23]. We model the search engine as a big database of the Web that logs in a file stored in its hosting server all queries of the user. This file is associated to the user's identifier from the cookies.

The Dissociating Privacy Agent is an agent, in the sense of a program that takes decisions without the user's supervision. It acts like a proxy between the user and the search engine and is implemented as an add-on for Firefox.

The strategy that the agent follows is based on the assumption that users are multifaceted individuals, meaning that they are interested in many different areas (e.g. "Science", "Sport" or "Music"). Thus, the identity of the user is formed by the union of the facets of his personality and this is what makes him unique among the population of all users.

In order to protect the identity of the user, the agent dissociates these facets by creating a new log for each one in the search-engine's server. As a result, the identity of the user is divided and it is more difficult to perform a re-identification by using the dissociated logs. Note that dissociated logs are still useful for profiling as, by construction, they preserve partial but real interests of the user that the search engine may derive for a later use in personalization.

In order to carry out dissociation we define several virtual identities each one of them owning a corresponding log at the server side. These virtual identities are built by generating new values for data that the server uses to track the users (e.g. identification numbers in cookies). Then, in conjunction with a proper management of the HTTP connections, we create a different session in the server for each virtual identity. Each of these sessions has a context involved that is formed by all the interaction of the user: jar of cookies, history of queries, history of clicked links, lists of results, among others (see Figure 1).

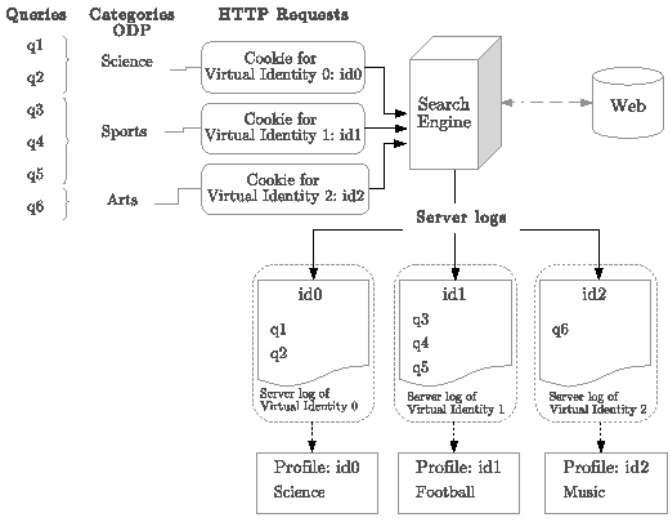

Fig. 1. Representation of the spoofing and dissociation of queries

A critical part of this strategy is classification of queries according to the user's facets. The user sends queries through the add-on's interface and the agent classifies them before submitting them to the search engine. DisPA uses the taxonomy of the Open Directory Project (ODP) for this purpose. The ODP is an index of the web that can be thought as a tree where nodes are categories and leafs are web pages. Behind the ODP there is a big community of editors that classify web pages into the ODP taxonomy adding some description and keywords.

The facets of the user are modelled as categories of the **first level** of the ODP tree which are: *Adult, Arts, Games, Shopping, Business, Health, Society, Computers, Home, News, Reference, Recreation, Sports, Science, Society*. In order to classify a query, we perform a faceted search in a local search engine that is built by creating an inverse index of the documents in the ODP corpus. The outcome of the search is a vector with coordinates the number of hits of the query in each of the categories. That is, let q be a query, given the set of categories $C = \{c_1, c_2, \ldots, c_n\}$, the vector v would be

$$v(q) := (h_1,\, h_2, \ldots, h_n) \tag{1}$$

where h_i is the number of documents indexed in category c_i hit by the query q, for $i = 1, \ldots, n$.

To solve the problem of polysemy, we also perform a local personalization to disambiguate queries that have different meanings. Our approach is based on the browser's history to extract a weight for each category. We consider the last k web pages that the user visited and classify them in the same local search engine. This way we obtain a vector of weights

$$w(q) := (w_1,\, w_2, \ldots, w_n)\,,$$

where the hits of the query and the past interests of the user are combined. Lastly, we build our classifier as

$$\text{classify}\,(q) = \underset{c_i \in C}{\arg\max}\, \{w_i\}\,. \tag{2}$$

More details about the classification algorithm can be found in [18] where it is fully specified.

B. User's specializations

One of the main flaws of this first implementation is that the set of categories used for classifying is fixed and does not take into account user's specializations. A user may have a lot of facets but he may be interested in ones much more than in others. For example, compare the queries that are sent from the computer at the work place and the ones that are sent from the computer at home.

In some cases, the interests of the user are specialized. Then, most of the queries are classified in one of the categories and the corresponding log is almost the whole original log. As an example to illustrate this, imagine that the user is very keen on computers. Then, queries fall mostly in the category *"Computers"* and the other categories are barely used. As a result, the dissociation by means of the first level of the ODP has no effect and the agent fails in its attempt to protect the user.

The main idea of our approach is that this category might be expanded to include more specific categories that describe user's interest more accurately. In the example above, computers would be expanded with the children of its node in the ODP tree: *AI, Algorithms, Games, Hacking, Internet, etc.* This way queries are more sparse and the dissociation is more effective.

Nevertheless, note that there is a trade-off between privacy and personalization in this process. Categories may range from very broad (upper levels of the tree), to very specific (lower levels), to the extreme of considering each individual query as a category. On the one hand, the former configuration would provide more personalization because we yield more data to the server, but it's obvious that there is more disclosure risk. On the other hand, the latter would provide less personalization because dissociated logs would contain only one query but the user would enjoy more privacy.

Besides, we have to consider long-term and short-term interests. A user specialization may be temporal and change with time. The system must be self-adaptive and rearrange the set of categories for classification automatically. For instance, in the example above, if the user becomes suddenly more interested in the topic *"Music"*, the system may retract the old category *"Computers"* and expand *"Music"*.

In the following section we are going to give a deeper explanation of this system and some details about the implementation.

III. Design

To implement the solution stated in the previous section we will expand or/and retract categories of the ODP tree whenever the number of queries in the categories is not balanced, i.e., there are categories that have too many queries. For this reason, one of the main challenges is to decide when this number is too large.

A. Coefficient of Variation

If we normalize the vector defined in expression 1, we may consider it as the distribution of probabilities of n random variables X_i representing the event of the query q belonging to the category c_i, for $i = 1, \ldots, n$.

Our approach is to measure the dispersion of this distribution and set a threshold that indicates when the number of queries per category is unbalanced and, thus, we have to expand or retract a category of the tree. We compute the coefficient of variation, which is a normalized measure of dispersion and is defined as

$$c_v := \frac{\sigma}{\mu},$$

where σ and μ are the the standard deviation and the mean of the distribution respectively. Without loss of generalization, we can consider that all categories start having one query already classified. This has no effect on the dispersion and we avoid definition problems assuring that μ is never equal to zero.

B. Self-adaptive classification

Let $C = \{c_1, \ldots, c_n\}$ be the set of categories used for classification referred in Section II-A. When the dispersion reaches a given threshold, we find which is the category with number of queries more deviated from the mean and look whether is a positive or a negative deviation. In the positive case, the category has a number of queries much higher than the mean and hence it has to be expanded.

The expansion operation is to add all the children of the category to be expanded into C. Note that we do not remove the parent from C because otherwise we would lose a possible outcome of the classification. For instance, imagine a query that hits documents contained exclusively in the parent.

In case that it is a negative deviation, the category has too few queries and it is not worth taking it into account. Thus, if this happens with all the categories that share the same parent, we can retract the parent and aggregate all them in one category.

Once we have some categories expanded, in order to classify a query using the classifier stated in the Expression 2, we perform a level-wise classification described in the Algorithm 1.

Algorithm 1 Level-wise classification

Require: A $mapping = \{C; w(q)\}$
Ensure: The label of the query: $maxCat$
  Initialization: $parent \leftarrow root$
  $maxCat \leftarrow$ "$Others$"
  $maxValue \leftarrow 0$
  repeat
    for all $entry$ **in** $mapping$ **do**
      $category \leftarrow entry.category$
      if $category$ **child of** $parent$ **then**
        $value \leftarrow entry.value$
        if $value > maxValue$ **then**
          $maxValue \leftarrow value$
          $maxCat \leftarrow category$
        end if
      end if
    end for
    $parent \leftarrow maxCat$
  until $maxCat$ **is not expanded**

Algorithm 1 shows that we begin in the first level of the tree and find the category of maximum weight. Then, if it has been expanded, we find the child with maximum weight and so on with the lower levels, until the category that maximizes the current level has not been expanded.

During the expansion, we generate a new virtual identity for each child and the virtual identity of the parent stays the same. This way, the log of the parent in the server may contain queries of other subcategories but it does not affect to personalization. From the point of view of disclosure risk we know that, now that the children is taken into account, the category will stop growing so much. When retracting a category we just use the virtual identity of the parent that we preserved in the expansion operation and, if we expand it again, we reuse the old virtual identities for the children.

C. Trade-off between personalization and privacy

Note that, as we mentioned in Section II-B, there is a trade-off between privacy and personalization that can be adjusted by the election of the set C. If we choose very specific categories, it is more difficult to personalize for the server, whereas if we choose general categories, we provide more data that the server can exploit, but disclosure risk increases.

Therefore, the continued expansion of categories could obstruct personalization. The problem is that search engine providers do not reveal which are the techniques used to personalize web searches. Moreover, personalization is a slow process and may take long time lapses to notice any effect. Consequently, the assessment of personalization turns to be one of the main problems that query classification deals with.

We cope with this limitation by adding two boundaries that limit the expansion and retraction of the tree. The upper boundary is the first level of the tree and the lower boundary is passed as a parameter. Once the lower boundary is defined we will not be able to expand a category that is in a level below of it.

As a result, we are able to adjust the level of sparseness of the logs in the server and, thereby, adjust the trade-off between privacy and personalization.

D. Algorithm

Every time that a query is sent, the agent follows Algorithm 2. This algorithm expands and retracts the more deviated categories in order to keep the Coefficient of Variation (CV) below the chosen threshold.

Note that we compute the CV upon an auxiliary set of categories. In order to make sure that the stop condition is always met, we remove those categories that are deviated but cannot be expanded due to the tree boundaries described before.

The function *findMoreDeviated()* takes a set of categories as an argument and returns the one with maximum deviation from the mean. The functions *expand()* and *retract()* work as it has been described in Section III-B. *remove()* and *getLevel()* are methods that remove a category from the list of categories and return the level of the category in the tree, respectively.

Algorithm 2 Self-adaptive classification

Require: Set of categories C
Ensure: Adapted set C'
  $computeCV(C)$
  $auxC \leftarrow C$
  while $CV > threshold$ **do**
    $devCat \leftarrow findMoreDeviated(C)$
    if $devCat.numQueries > mean(C)$ **then**
      if $devCat.getLevel() > lowestLevel$ **or** $devCat$ **is expanded or** $devCat$ **has no children then**
        $auxC.remove(devCat)$
      else
        $expand(devCat)$
      end if
    else
      if $devCat.getLevel() \leq uppermostLevel$ **then**
        $auxC.remove(devCat)$
      else
        $retract(devCat)$
      end if
    end if
    $computeCV(auxC)$
  end while
  $C' \leftarrow auxC$

IV. Empirical results

In order to test the agent and prove that disclosure risk is reduced by the enhancement proposed in this article, we used the linkage algorithm described in [18]. This algorithm is supposed to be applied by an adversary on the logs in the

server in order to link those that have been dissociated by DisPA, and rebuild the original log of the user.

A. Adversary model

We assume that the adversary is the search engine provider or a third party that has access to all the logs in the server. The goal of the adversary is to extract new information of the user from his logs or, in the best case from the adversary's point of view, discover the identity of the user. In the following lines we will describe the capabilities of the adversary, which are limited because we assume that the adversary lacks of background knowledge.

The attack carried out by the adversary exploits terms that appear in most of dissociated logs of the user but that are very uncommon among the population of all users. These terms will help the attacker to tell apart between the victim's logs and the others'.

B. Attacking algorithm

Firstly, the algorithm expresses logs as vectors using a tf-idf scheme. The rationale is that tf-idf reflects the importance of a term in a log offset by its frequency in the collection of all logs. Then, this algorithm clusters the vectorial space using the DBSCAN algorithm with the cosine similarity as a distance.

The linkage algorithm starts with one or more seeds corresponding to logs that are known to be of the user. At the end, all clusters that contain a seed are joined into one unique cluster that represents the original log.

We use the same measure as in [18] for this evaluation. We consider the binary classification defined by the property of a query being part of the final cluster or not. We also consider the F1-Score as the measure of the disclosure risk. The F1-Score is a measure to assess the quality of the clustering that combines precision and recall. Note that in our case, true positives are queries of the target user that fall in the final cluster, false positives are queries of other users that fall in the final cluster, true negatives are queries of other users that fall out of the final cluster and false negatives are queries of the target user that fall out of the final cluster.

Besides, the DBSCAN clustering needs a parameter as an input that defines the neighborhood of a cluster. Despite the fact that this parameter is not known a priori by the attacker we want to evaluate the worst case. For this reason we decided to take a range of possible values and assume that the attacker knows which one is the optimal.

C. Experiments

For the empirical part we used the AOL released dataset but, as we were also interested in very specialized users, we developed a generator of queries based on the keywords stored in the ODP that we referred in Section II-A. The generator takes a distribution of probabilities for the set of categories as a parameter and generates a log of queries according to them.

For the first experiment we created a log choosing the following distribution

$Adults$	0
$Arts$	0
$Games$	0.02
$Reference$	0.02
$Shopping$	0
$Business$	0.04
$Health$	0.02
$News$	0
$Society$	0.1
$Computers$	0.8
$Home$	0
$Science$	0
$Sports$	0

and generated a set of 300 queries.

For the first experiment we simulated the submitting of these queries first using the current version of DisPA and, later on, the extended version presented in this article with the self-adaptive classification setting the lower boundary to three levels and the threshold of the coefficient of variation to 80%.

We added 20 random users from the AOL released dataset and applied the linkage algorithm. We considered the worst case in which the attacker knows that the biggest log certainly belongs to the target user. Afterwards, we did the same but using the agent with the self-adaptive system.

In order to decide if the user is protected or not, we set 50% of disclosure risk as a threshold. If the F1-Score is below this 50% we say that the user is protected whereas, if it is higher, we say that the user is not protected.

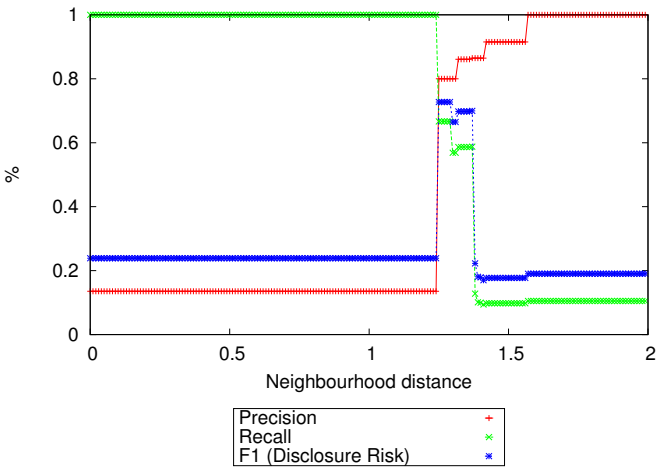

Fig. 2. Standard DisPA

In the Figure 2 we show the results of the first part of the experiment. As we see, there are some values of the neighbourhood distance for which the user is not protected because the disclosure risk (i.e. F1-Score measure) is above the 50%. We can see that it makes no sense to go on evaluating for values greater than 2 because the precision is maximum. This means that all logs of the target user fall in the final

cluster and it will not improve. In fact, we see that all logs fall in the final cluster since recall is very low.

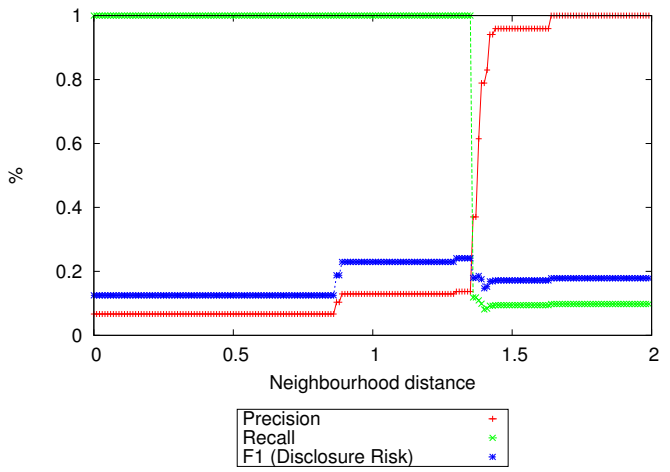

Fig. 3. Self-adaptive classification

In the second part of the experiment we did exactly the same, although the seeds changed because we were considering a different collection of dissociated logs. In fact, two categories were expanded during the simulation: "$Top/Computers$" and "$Top/Computers/Internet$".

As we see in Figure 3, the disclosure risk is below the 50% and, therefore, the user is protected. The percentage of disclosure risk reduction from the standard DisPA in the worst case is around 67%.

We designed a second experiment under the same assumptions but using the log of a real user. In this case we used Thelma Arnold's log, the first user of the AOL released dataset that was discovered by the journalist of the NYT [6].

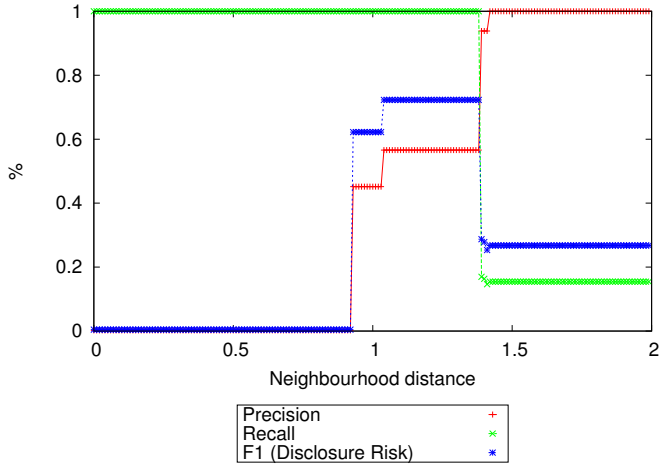

Fig. 4. Standard DisPA

In Figure 4 we show the results for the case using the standard DisPA for an election of 20 random users an a random seed. In this case the user is not protected, since the are values of the neighbourhood distance for which the disclosure risk is above the 50%

In the second part of this experiment we used the self-adaptive approach and we obtained the results shown in Figure 5.

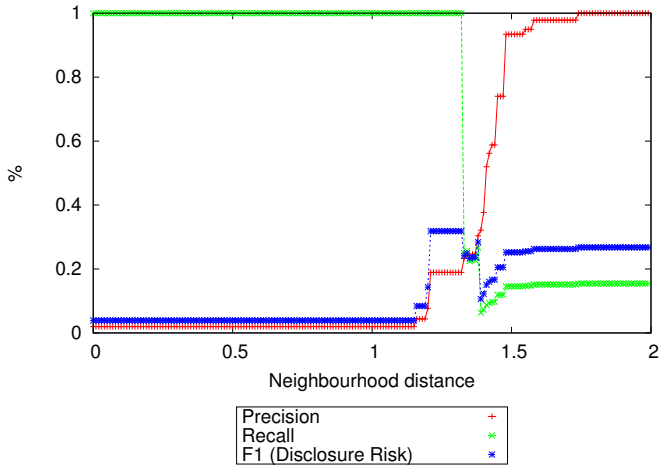

Fig. 5. Self-adaptive classification

Now the user is protected but the percentage of reduction of disclosure risk is not so steep. This time this percentage is around the 56%. This may be because the profile of a real user like Thelma Arnold, using the computer in her house, is not as specialized as the profile artificially generated before.

D. Performance

The overhead in performance is reasonable according to our estimations using Google's search engine. The response time in the worst case (when DisPA has to create a new identity for the category) is around 2.5 seconds, approximately five times more than a direct search. After this bootstrapping process, the normal case is that the identity already exists, in that case the average response time is 1.5 seconds (three times the response time in a standard search). Obviously, if the query has already been sent previously, the results can be fetched from the cache and the response time is negligible.

Despite having an average overhead of 3 times the standard search,a user may consider that 1 second extra is not such a long time to expend in exchange of a more protected search.

V. Future Work

Empirical results have proved that disclosure risk is much lower with this method. However, the self-adaptive classification has some drawbacks that future research may deal with.

One of the most important shortcoming is that we do not have measures for personalization and we are forced to define a lower bound for the expansion of the tree. Further research could find functions to assess personalization and, thereby, provide a method to automatically limit this expansion.

Another line of future work, could assess how short-term interests are gathered by our approach and also, an analysis of the sensibility of the threshold for the coefficient of variation.

Besides, the linkage algorithm used in the experiments could be improved in several ways. Observe that the main objective of the algorithm is to provide a way to, given a set of logs, determine if they belong to the same user or not. In the evaluation part of this work, we follow the approach of simulating the server of the search engine by adding some random logs. This way we approximate the frequency of terms in the corpus. A statistical analysis should be carried out on a real server in order to incorporate this information to the algorithm and obtain a more accurate evaluation. Note that this analysis should be done periodically since these frequencies change over time. Furthermore, this would allow us to work exclusively with the logs of the target user.

Secondly, there could be used other clustering algorithms like OPTICS, which addresses the inconvenient of defining a neighbourhood distance and takes into account clusters of different densities.

Lastly, sequential clustering techniques like the ones described in [24] could be used to reduce the complexity of the algorithm.

VI. Conclusion

The main contribution of this research is the implementation of an improvement for the Dissociating Privacy Agent that provides less disclosure risk in web search. This enhancement overcomes the problems associated to user's specializations such as users that are interested in very specific areas of knowledge.

Moreover, we proved that disclosure risk was improved for our own linkage algorithm and compared the results between a generated profile intentionally unbalanced to one category of interest and a real user from the AOL's released dataset.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 262608.

Partial support by the Spanish MEC projects ARES (CONSOLIDER INGENIO 2010 CSD2007-00004), eAEGIS (TSI2007-65406-C03-02) and COPRIVACY (TIN2011-27076-C03-03) is acknowledged.

References

[1] S. Hansell, "Increasingly, Internet's Data Trail Leads to Court," Feb. 2006, New York Times.

[2] P. Norvig, "Search Algorithms with Google Director of Research Peter Norvig," Oct. 2011, Stone Temple Consulting.

[3] M. Speretta and S. Gauch, "Personalized search based on user search histories," in *Proceedings of the 2005 International Conference on Web Intelligence*. IEEE, 2005, pp. 622–628.

[4] EFF, "AOL's Massive Data Leak," 2009.

[5] G. Sadetsky, "AOL Data," Aug. 2006. [Online]. Available: http://www.gregsadetsky.com/aol-data/

[6] M. Barbaro and T. Zeller, "A Face Is Exposed for AOL Searcher No. 4417749," 2006, New York Times.

[7] J. Domingo-Ferrer, A. Solanas, and J. Castellà-Roca, "h(k)-Private Information Retrieval from Privacy-Uncooperative Queryable Databases," 2008.

[8] D. Howe and H. Nissenbaum, "TrackMeNot: Resisting surveillance in web search," *Lessons from the Identity Trail: Anonymity*, 2009.

[9] J. Domingo-Ferrer, M. Bras-Amorós, Q. Wu, and J. Manjón, "User-private information retrieval based on a peer-to-peer community," *Data and Knowledge Engineering*, 2009.

[10] B. Shapira, Y. Elovici, A. Meshiach, and T. Kuflik, "PRAW: A PRivAcy model for the Web," *Journal of the American Society for Information Science and Technology*, vol. 56, no. 2, pp. 159–172, Jan. 2005.

[11] D. Rebollo-Monedero and J. Forne, "Optimized Query Forgery for Private Information Retrieval," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4631–4642, Sep. 2010.

[12] S. Ye, F. Wu, R. Pandey, and H. Chen, "Noise Injection for Search Privacy Protection," *2009 International Conference on Computational Science and Engineering*, pp. 1–8, 2009.

[13] M. Murugesan and C. Clifton, "Plausibly deniable search," *Workshop on Secure Knowledge Management*, vol. 1, pp. 3–8, 2008.

[14] T. Paul, M. Stopczynski, D. Puscher, M. Volkamer, and T. Strufe, "C4PS - Helping Facebookers Manage Their Privacy Settings," in *Social Informatics*, ser. Lecture Notes in Computer Science, K. Aberer, A. Flache, W. Jager, L. Liu, J. Tang, and C. Guéret, Eds. Springer Berlin Heidelberg, 2012, pp. 188–201.

[15] L. Fang and K. LeFevre, "Privacy wizards for social networking sites," in *Proceedings of the 19th international conference on World wide web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 351–360.

[16] T. Paul, D. Puscher, and T. Strufe, "Improving the Usability of Privacy Settings in Facebook," *CoRR*, vol. abs/1109.6, 2011.

[17] J. Becker and H. Chen, "Measuring privacy risk in online social networks," in *Web 2.0 Security and Privacy (W2SP)*, Oakland, CA, 2009.

[18] M. Juárez and V. Torra, "Toward a privacy agent for information retrieval," *International Journal of Intelligent Systems*, vol. 28, pp. 606–622, June 2013.

[19] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *in proceedings of the 36th Annual Symposium on Foundations of Computer Science*, IEEE. IEEE Comput. Soc. Press, 1995, pp. 41–50.

[20] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single database, computationally-private information retrieval," *in proceedings of the 38th Annual Symposium on Foundations of Computer Science*, 1997.

[21] E. Balsa, C. Troncoso, and C. Diaz, "OB-PWS: Obfuscation-Based Private Web Search," *Security and Privacy (SP), 2012*, 2012.

[22] Google, "Key Terms - Policies and Principles," Apr. 2012.

[23] V. Toubiana and H. Nissenbaum, "Analysis of Google Logs Retention Policies," *Journal of Privacy and Confidentiality*, vol. 3, no. 1, 2011.

[24] S. Miyamoto and K. Arai, "Different sequential clustering algorithms and sequential regression models," in *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*. IEEE, 2009, pp. 1107–1112.