International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015
ISSN 2091-2730

# Refinder Web Search

Shyam Sunder Dusadh[1], Shafiqe Ahmad[2], S.Pothumani[3]

[1,2]U.G Student, Dept.of.CSE, Bharath University, Chennai, India
[3]Assistant Professor, Bharath University, Chennai,India

[1]officialshyam@yahoo.in [2] khanshaf09@gmail.com

**Abstract**— Personalized Web Search provide better search result as per user queries. In the existing system the search engine will retrieve the irrelevant results for the user query. In the proposed model the query will be processed based on Personalized Web Search. The query will be retrieved based on Click Log based and Profile based. The Profile based retrieval will be done through browsing history, click through data and bookmarks. We are purposing an addition method to Track the Lead Time of the Resultant Website & Obtaining the Feedback of the user for any URL. Based on Lead time, Feedback and above said Proposed Methods we would Rank the best Resultant for any Users.

**Keywords:** Refinder, Lead Time, Click Log, Personalized Web Search, greedyDP, greedyIL, prune-leaf

## 1. INTRODUCTION

There are lots of search engines available on web to get any type of information. But not all of them provides good results, instead it returns some irrelevant result that does not matches the user queries. To solve this type of problem Personalized web search is helpful. Personalized Web Search (PWS)[1] is a general search engine technique which aimed to provide better search results as per individual user queries. PWS can generally be categorized into two types, first is click-log-based methods and second is profile-based[1].

Click-log based methods simply impose bias to clicked pages in the user's query history. It can only work on repeated queries from the same user. However profile-based method used to improve search experience and it also generates complicated user interest models from user profiling techniques. Although both of them has some limitation in click-log based as well as profile based. Click-log based strategy can only work on repeated queries from the same user[2], which is a strong limitation confining its applicability and profile based method is unstable under some circumstances.

So, we are purposing an addition method to rank the best resultant for any user. In this method we are going to track the Lead Time of the Resultant Website & will obtain the Feedback of the user for any URL and then according to the lead time and feedback of the user it will re-rank the previous results and provide best results as per user queries.

The rest of this paper is organized as follows: Section 2 reviews the related work, focusing on PWS and its privacy preservation. Section 3 presents the architecture diagram of the project. Section 4. Gives a description about modules used in this paper. Section 5. Presents the generalisation algorithms for the project. Section 6 demo of the project (screen shot) Section 7. Concludes the paper.

## 2. PREVIOUS WORS

### 2.1 Supporting Privacy Protection in Personalized Web Search

Personalized web search provide effective search queries for user. But research says that user doesn't want to disclose or share his/her personal information. To solve this problem author purposes a PWS framework called UPS. UPS (User Customizable Privacy-Preserving Search) can generalize profiles by queries and it's also protect user's specified privacy requirements. UPS contains of a search engine server and a number of clients. Each client (user) approaching the search service trusts no one but himself/herself. The major component for privacy protection is an online profiler implemented as a search proxy running on the client side machine itself. The proxy observes both the complete user profile, in a picking order of nodes with semantics, and the user-specified (customized) privacy requirements represented as a set of sensitive-nodes.

UPS is differentiated from conventional Personalised Web Search in such a way that it Provides runtime profiling, which is effective to optimizes the personalization utility with user's privacy Requirements and it allows for customization of privacy needs and even it does not require any iterative user interaction.

UPS is used for runtime generalization. Runtime generalization help to make balanced between two prognostic metrics that evaluate of personalization and the privacy risk to bring out the generalized profile.

For runtime generalization they presented two greedy algorithms first one is GreedyDP and second one GreedyIL and both algorithm is efficient to produce best results for personalized user query.

In our purposed system we are adding lead time techniques and feedback of the user to enhance the personalized web search by re-ranking the result on the basis of lead time and feedback of the user.

**2.2 Profile-Based Personalization**

In the Previous works on profile-based PWS they mainly focus on improving the search utility. The principle idea of these works is to adapt the search results by referring to, often implicitly, a user profile that bring out a particular information goal.

There are many profile representations that available in the literature to alleviate different personalization strategies. Earlier techniques apply term lists/vectors or bag of words to represent their profile. However, the recent works create profiles in hierarchical structures because of their stronger descriptive ability, higher access efficiency, and better scalability. The majority of the hierarchical representations are made with existing weighted topic hierarchy/graph, such as ODP[1], Wikipedia[2] and so on. In our purposed system we do not aim on the implementation of the user profiles. Actually we are enhancing the supporting PWS by using re-ranking methods and make it user friendly by providing feedback option.

In the literature the performance measures of PWS, Normalized Discounted Cumulative Gain (nDCG) is a common measure of the effectiveness of an information retrieval system. It is works on an ungraded relevance scale of item-positions in the result list.

Meanwhile our work is to enhance the PWS so we are using click-log and profile based technique to produce relevant search while lead time technique and feedback are used to enhance the PWS.

**2.3 Privacy Protection in PWS System**

There are two classes of privacy protection problems for PWS.

- First one class includes those treat privacy as the identification of an individual.
- The other hold those consider the sensitivity of the data, related to the user profiles, exposed to the PWS server.

1. Open Directory Project (ODP), http://dmoz.org/.

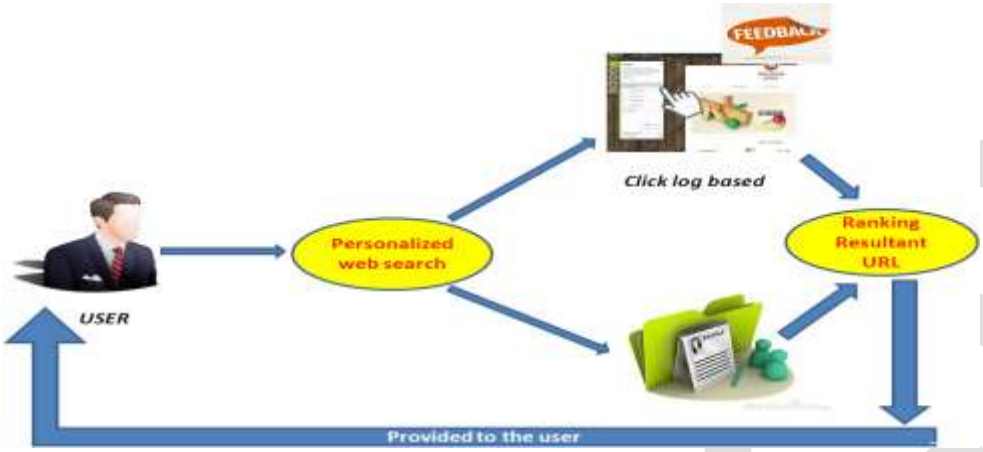
2. Wikipedia, the Free Encyclopaedia, http://www.wikipedia.org/

The key works in the literature of protecting user identifications (class one) try to solve the privacy problem on different levels, including the group identity, the pseudo identity, no personal information, and no identity. The first level solution is proved to fragile. The third and fourth levels are visionary because of their high cost in cryptography and communication. So, the existing efforts focus on the second level. [8] and [9] generating a group profile of k users to provide online anonymity on user profiles. Using this approach, the linkage between the query and a single user is broken, the useless user profile (UUP) protocol is implemented to shuffle queries among a group of users who issue them that result any queries cannot profile a certain individual. In this scheme, every user acts as a search agency of his or her neighbours. users can decide to submit the query on behalf of who issued it, or forward it to other neighbours. The shortcomings of current solutions in class one is the high cost introduced due to the collaboration and communication.

The solutions in class two do not require third-party assistance or collaborations between social network entries. In this solution, users only trust themselves and cannot tolerate the exposure of their complete profiles an anonymity server. Horvitz and Krause employ statistical techniques to learn a probabilistic model, and then this model is used to generate the near-optimal partial profile. The main limitation in this work is that probabilistic model is trained through predefined frequent queries and it builds the user profile as a finite set of attributes. The literature proposed a privacy protection solution for PWS based on hierarchical profiles. User-specified threshold

is used to generalize profile to obtain in effect as a rooted sub tree of the complete profile. But this work does not address the utility of query, which is most important for the service quality of PWS.

## 3. ARCHITECTURE DIAGRAM




## 4. MODULES

### 4.1 USER QUERY

Query deployment module is used to create the Search Bar by which the User will enter the query. We are going to implement this project as web project. We will load our project into the web server and then we will execute this project. To create user query interface page, we'll develop this page using web based coding like Java Server Pages and Servlets. This User interface page will be connected to the backend database in which we will have the dataset. So that the result will be retrieved back to the user for their enter query.

### 4.2 SERVER

The server will maintain the database which consists of the large amount of data from which the exact result will be retrieved for the user surfed query. The server will also retrieve the data based on the search option the user is wanted to search of the information. The server will also implement some techniques while retrieving the data from the database.

### 4.3 CLUSTERING THE DOCUMENTS

In this module we are clustering the named documents. The Clustering will be processed as per name or keyword or tag values in the documents. So that we can cluster the documents named documents effectively. Since we are applying the clustering mechanism we can extract the best match result with maximum probability.

### 4.4 STEMMING ALGORITHM

Stemming algorithm is used to filter the stopping words from the query and search for the exact result that the user is surfing for. So that we can retrieve best matched results from the server's database. We will use word stemmer algorithm to filter the stopping words from the query.

### 4.5 SCORING ALGORITHM

In this module we are going to apply the Ranking algorithm to rank the result as documents weights. For an example if the user enter the query as "Cloud Computing" then the server will retrieve the data and order them according to the document weights. Document will be ranks using the below mentioned formulae.

Document weight= Total of query word query present in the document / Total number words in the document.

### 4.6 TOP K QUERY AND DATA RETRIVAL

In this module we can retrieve the documents using Top K query algorithm. By using this algorithm we can retrieve the Top K best matched results for the user entered query. SO that we retrieve most matched documents for the entered query.

## 5. THE GENERALIZATION ALGORITHM

At first we introducing a brute-force optimal algorithm, then we implement two Greedy algorithms, namely the GreedyDP and GreedyIL.

### 5.1 THE BRUTE-FORCE ALGORITHM

The brute-force algorithm exhausts all possible rooted sub trees of a given seed profile to find the optimal generalization. The requirements are respected during the exhaustion. The sub tree with the optimal utility is chosen as the result. Although the seed profile H0 is significantly smaller than G, the exponential computational complexity of brute-force algorithm is still not acceptable. Formally, we have the following theorem whose proof is given in the link http://doi.ieeecomputersociety.org/10.1109/TKDE.2012.201.

### 5.2 THE GREEDYDP ALGORITHM

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

The first greedy algorithm GreedyDP works in a bottom up direction. Starting from G0, in every $I^{th}$ iteration, GreedyDP chooses a leaf topic $t \in T_g$, (q) for pruning, trying to maximize the utility of the output of the current iteration, namely $G_{i+1}$.During the iterations, we maintain a best profile-so-far, which indicates the $G_{i+1}$.having the highest discriminating power while satisfying the risk constraint. The process terminates when the profile is generalized to a root-topic. Which is the best-profile-so-far will be the final result ($G_0$) of the algorithm.

The main problem of GreedyDP is that it needs re-computation of all candidate profiles (together with their discriminating power and privacy risk) generated from attempts of prune-leaf on all $t \in T_g$, (q) This causes significant memory requirements and computational cost.

### 5.3 THE GREEDYIL ALGORITHM

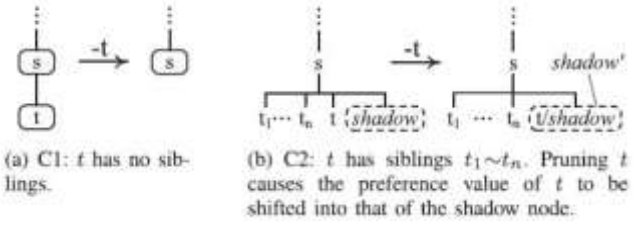
The GreedyIL algorithm implement to improve the efficiency of the generalization using heuristics based on several findings. The important finding is that any prune-leaf operation reduces the discriminating power of the profile. i.e, the DP displays monotonicity by prune-leaf. We have the following theorem:

**Theorem 1**. If G0 is a profile obtained by applying a prune-leaf operation on G, then $DP(q,G) \geq DP(q,G')$.

Considering operation $G_i$ -t $\rightarrow$ $G_{i+1}$. in the Ith iteration, maximizing DP(q,$G_{i+1}$) is equivalent to minimizing the incurred information loss, which is defined as DP(q, $G_i$ ) -DP(q, $G_{i+1}$)

The above finding motivates us to maintain a priority queue of candidate prune-leaf operators in descending order of the information loss caused by the operator. Each candidate operator in the queue is a tuple like OP= < t, I L ( t, $G_i$) > , where IL( t, $G_i$) indicates the IL incurred by pruning t from $G_i$ and t is the leaf to be pruned by op. This queue, denoted by Q, enables fast retrieval of the best so-far candidate operator.

This Theorem also leads to the following heuristic, which helps to reduces the total computational cost Significantly




(a) C1: $t$ has no siblings.

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

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

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

Next finding is that the computation of IL can be simplified to the evaluation of ΔPG(q.G)= PG( q, $g_i$)-PG( q, $G_{i+1}$).The reason is that, referring to (12), the another term TS(q, G) remains unchanged for any pruning operations until a single leaf is left (in such case the only choice for pruning is the single leaf itself). Now we consider two possible cases as being showed in Fig. (C1) t is a node with no siblings, and (C2) t is a node with siblings. The case C1 is easy to handle. while, the evaluation of IL in case C2 requires introducing a shadow sibling of t. Each time if we try to prune t, we actually merge t into shadow to get a new shadow leaf shadow 0, together with the preference of t, i.e

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

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

**Heuristic 2**

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

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

The third finding is that, in case C1 already described above, prune-leaf only works on a single topic t. Thus, it does not affect the IL of other candidate operators in Q. In case C2, pruning t incurs re-computation of the preference values of its sibling nodes. Thus, we have

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

Algorithm 1 shows the pseudo code of the GreedyIL algorithm. Generally, GreedyIL traces the information loss of the discriminating power. This help to saves a lot of computational cost. In the Heuristic 1(line 5) avoids unimportant iterations. Heuristics 2 (line 4, 10, 14) further simplifies the computation of IL. Finally, Heuristics 3 (line 16) reduces the need for IL-re-computation significantly. All topics in the seed profile have sibling nodes In the worst case condition, then GreedyIL has computational complexity of O(|G0| *T|G0 (q)|). However, this is extremely extinct in practice. Therefore, GreedyIL is expected to significantly outperform GreedyDP.

ALGORITHM 1: GreedyIL

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

```
1  let Q be the IL-priority queue of prune-leaf decisions;
       i be the iteration index, initialized to 0;
   // Online decision whether personalize q or not
2  if DP(q, R) < μ then
3      Obtain the seed profile G_0 from Online-1;
4      Insert ⟨t, IL(t)⟩ into Q for all t ∈ T_H(q);
5      while risk(q, G_i) > δ do
6          Pop a prune-leaf operation on t from Q;
7          Set s ← par(t, G_i);
8          Process prune-leaf G_i --t--> G_{i+1};
9          if t has no siblings then                    // Case C1
10             Insert ⟨s, IL(s)⟩ to Q;
11         else if t has siblings then                  // Case C2
12             Merge t into shadow-sibling;
13             if No operations on t's siblings in Q then
14                 Insert ⟨s, IL(s)⟩ to Q;
15             else
16                 Update the IL-values for all operations on
                   t's siblings in Q;
17         Update i ← i + 1;
18     return G_i as G*;
19 return root(R) as G*;
```

## 6. SCREEN SHOTS

**Image 1:** Admin Page only access by admin

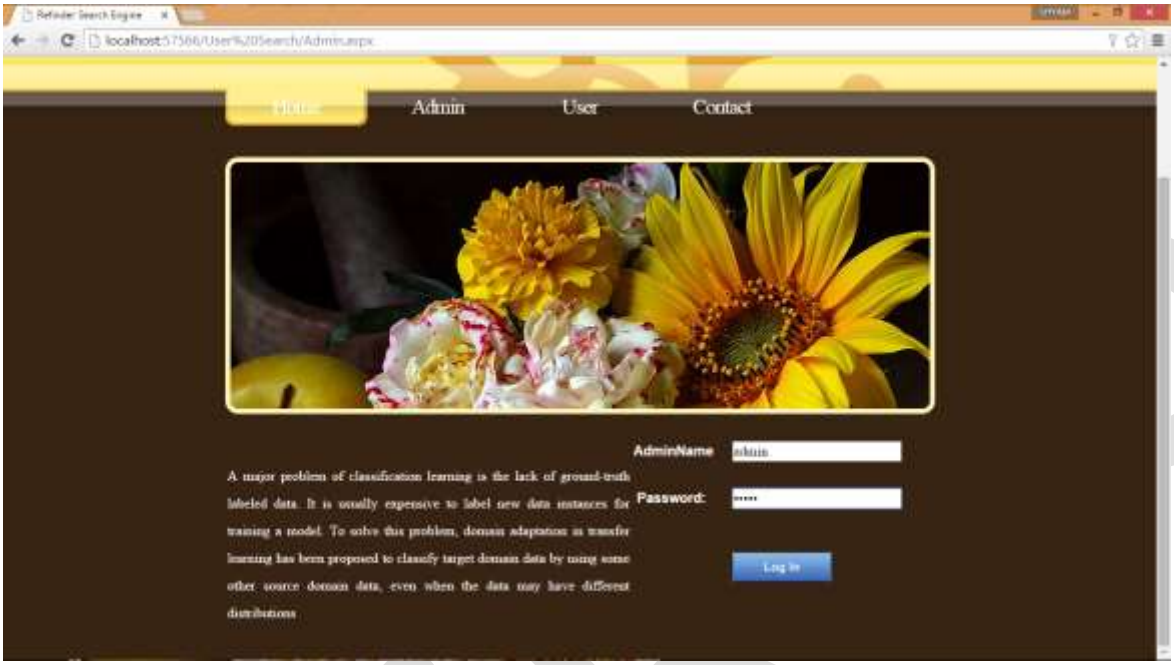



**Image 2:**web page after log in by admin


From this page admin control the search engine and from this page we can add new user, add web information, uploads files.

**Image 3:** sign up page for new user

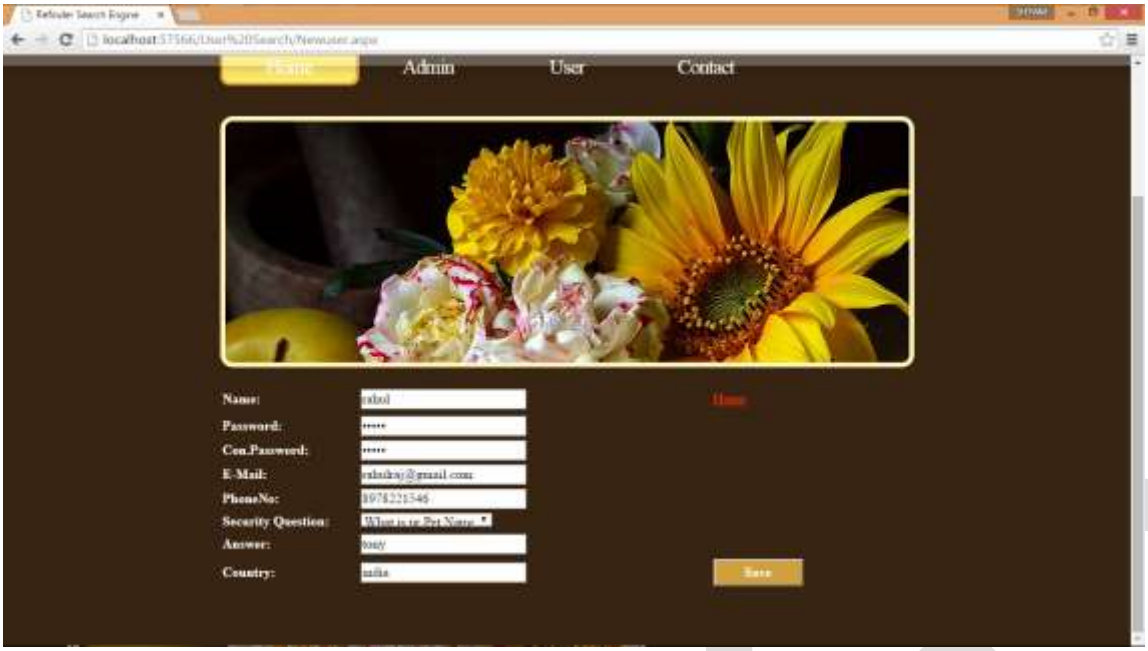



**Image 4:** log in page for user


**Image 5:** personalised web search page for user


**Image 6:** query result page for user

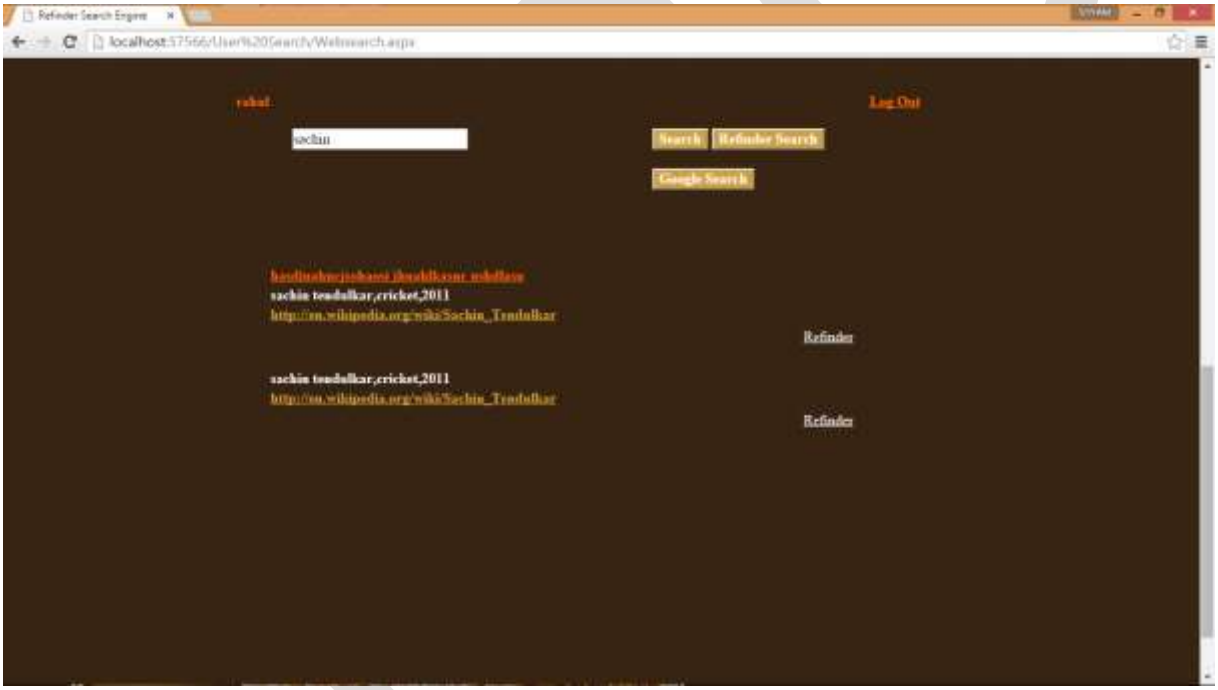



**Image 7:** refinder option to vew this result in future. It have three catogary date,location and activity

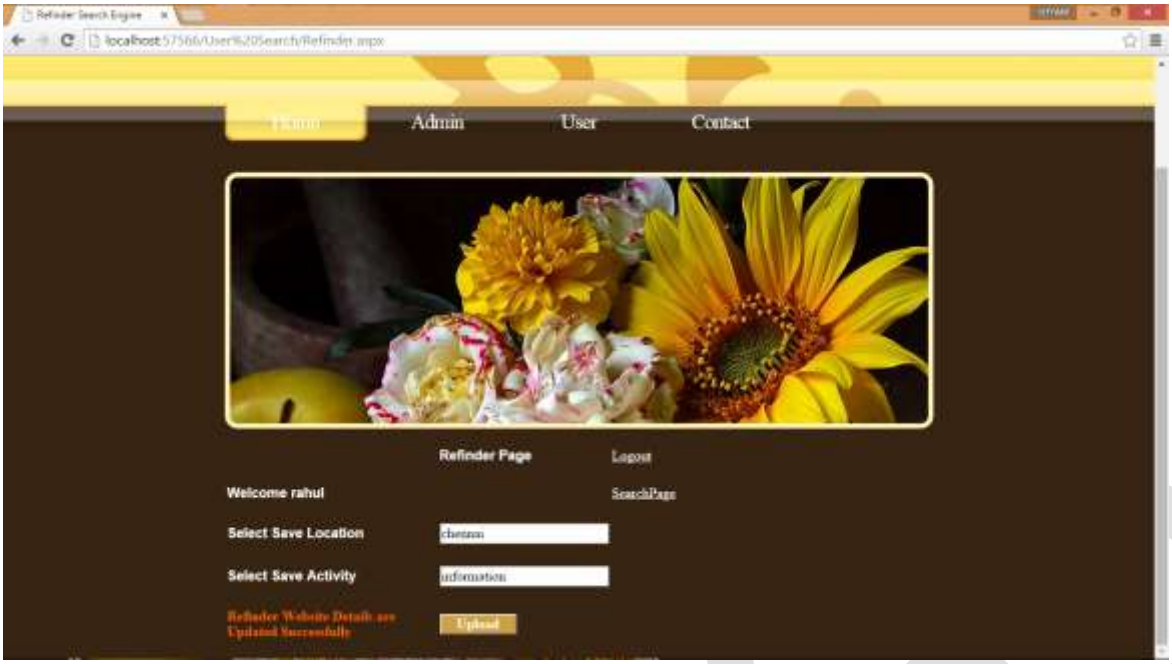
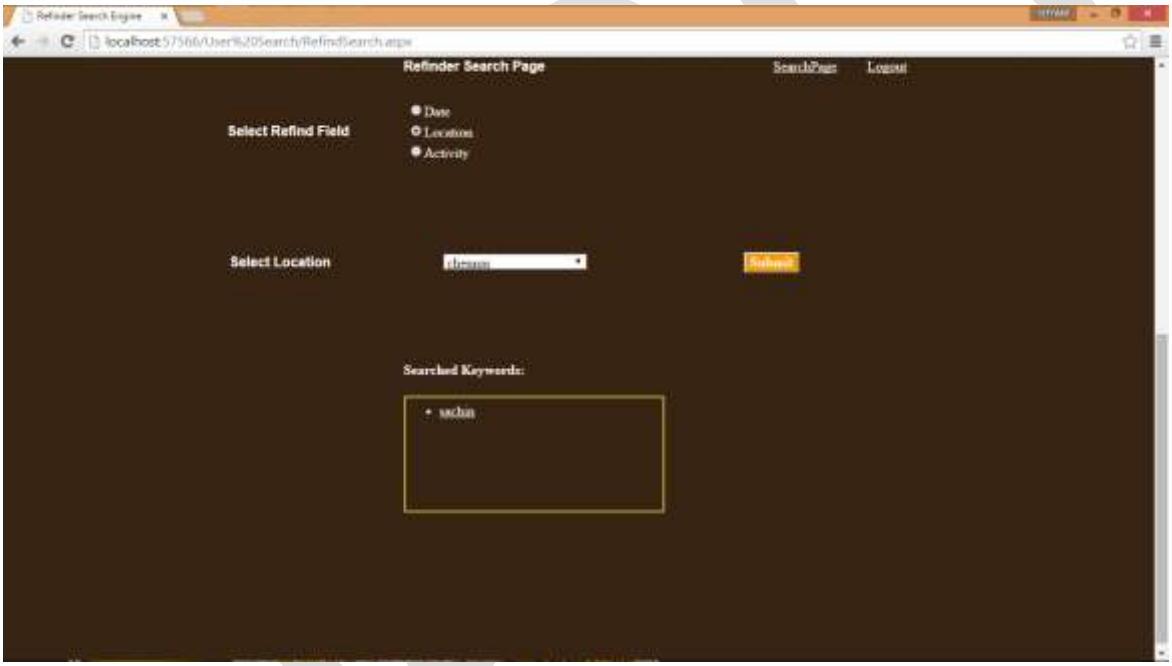



**Image 8:** with help of refinder search previous works can easily access by user anytime and anywhere.




## 7. ACKNOWLEDGMENT

The author would like to thank the Vice Chancellor, Dean-Engineering, Director, Secretary, Correspondent, and HOD of Computer Science & Engineering, Dr. K.P. Kaliyamurthie, Bharath University, and Chennai for their motivation and constant encouragement. The author would like to specially thank **Dr. A. Kumaravel**, **Dean , School of Computing**, for his guidance and for critical review of this manuscript and for his valuable input and fruitful discussions in completing the work and the Faculty Members of Department of Computer Science &Engineering. Also, he takes privilege in extending gratitude to his parents and family members who rendered their support throughout this Research work.

## 8. CONCLUSION

This paper presented an enhancement of privacy protection framework for personalized web search. The framework allowed users to specify customized privacy requirements via the hierarchical profiles. In addition, we add refinder option for user. When user search

queries, refinder option available in all result queries. If user get some useful information according to the query he can easily use refinder option to view these result again. Refinder sorted in three option 1.date 2.location 3.activity.

In date it short the result by date user use the refinder option for significant result. While in the location category it sort the result by location the user search the query. In activity category, refinder sot the result in the basis of what user take the action with that result.

For future work, we will try to make it more protected with one time password and it resist the adversaries with broader background knowledge or capability to capture a series of queries from the victim.

**REFERENCES:**

[1] Lidan Shou, He Bai, Ke Chen, and Gang Chen, "Supporting Privacy Protection In Personalized Web Search",proc. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, pp. 453-467,2012

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

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

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

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

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

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

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

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