can effectively adapt the retrieval function of a meta-search engine to a particular group of users, outperforming Google in terms of retrieval quality after only a couple of hundred training examples.

The differences between existing works and ours are

1) Most existing location-based search systems, require users to manually define their location preferences (with latitude-longitude pairs or text form), or to manually prepare a set of location sensitive topics. OBPMSE profiles both of the user's content and location preferences in the ontology based user profiles, which are automatically learned from the clickthrough and GPS data without requiring extra efforts from the user.
2) We propose and implement a new and realistic design for OBPMSE. To train the user profiles quickly and efficiently, our design forwards user requests to the PMSE server to handle the training and reranking processes.
3) Existing works on personalization do not address the issues of privacy preservation. OBPMSE addresses this issue by controlling the amount of information in the client's user profile being exposed to the OBPMSE server using two privacy parameters, which can control privacy smoothly, while maintaining good ranking quality.

3. System Design

Fig. 1 shows OBPMSE‟s client-server architecture, Which meets three important requirements. First, computation-intensive tasks, such as RSVM training, should be handled by the OBPMSE server due to the limited computational power on mobile devices. Second, data transmission between client and server should be minimized to ensure fast and efficient processing of the search. Third, click through data, representing precise user preferences on the search results, should be stored on the OBPMSE clients in order to preserve user privacy.

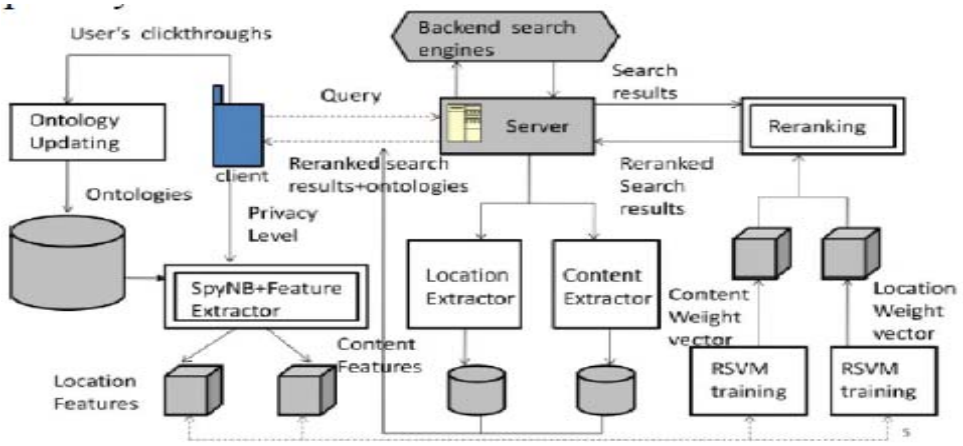

Figure 1: OBPMSE Client-Server Architecture

In the OBPMSE‟s client-server architecture, OBPMSE clients are responsible for storing the user clickthroughs and the ontologies derived from the OBPMSE server. Simple tasks, such as updating clickthoughs and ontologies, creating feature vectors, and displaying reranked search results are handled by the OBPMSE clients with limited computational power. On the other hand, heavy tasks, such as RSVM training and reranking of search results, are handled by the OBPMSE server. Moreover, in order to minimize the data transmission between client and server, the OBPMSE client would only need to submit a query together with the feature vectors to the OBPMSE server, and the server would automatically return a set of reranked search results according to the preferences stated in the feature vectors. The data transmission cost is minimized, because only the essential data (i.e., query, feature vectors, ontologies and search results) are transmitted between client and server during the personalization process. OBPMSE‟s design addressed the issues: 1) limited computational power on mobile devices, and 2) data transmission minimization.

3.1 Problem Definition

In the existing system there is only the query based searching is available, by using this query based mobile searching, it is not possible to extract the data from the search engine. Problems encountered in searching are exaggerated further when search engine users employ short queries. They cause relevant information to be missed if the query does not contain the exact keywords occurring in the documents. For these reasons, users face a difficult battle when searching for the exact documents and products that match their needs. Mobile web search introduces new challenges not present in traditional web search. We are using the ranking based searching and the GPS location based searching, by using these two we can easily extract the user query from the search engine.

3.1.1 System Overview

1) OBPMSE Client

The OBPMSE client is responsible for forwarding the search query to the server. Once the query is forwarded , the client is also responsible for accessing the search engine and retrieving the results. In this client-server architecture, clients are responsible for storing the user click throughs. Simple tasks, such as updating click throughs, creating user profiles, and displaying re-ranked search results are handled by the clients with limited computational power. Also, mobile client will not transmit the personal information to the server. The data transmission cost is minimized, because only the essential data (i.e., query and search results) are transmitted between client and server during the personalization process. Heavy tasks, such as training and re-ranking of search results, are also handled by the client. It consists of two major activities: 1) Re-ranking the search results at the client,

and 2) Clickthrough collection and updation at a mobile client.

2) Content and Location Search:

Once the search keyword is provided by user, the client asks for two options namely,

i. Web search

ii. Places search

Based on the option selected by the user, the corresponding search is done.

i. Web search:

This module is done for content-based searches. When this option is selected by the user, the general results from the google server are returned to the user's mobile. Back-end process: We use Custom Search API to retrieve the results and apply our re-ranking algorithm using the clickthrough data. Also the title, link and snippets are separately fetched from each result and displayed in our result page.

ii. Places search

This module is done for location-specific searches. For some searches where the results must be returned based on the particular location (eg. restaurant, school, hospital etc.) this option is selected by the user.

Back-end process: Once this option is selected, the latitude-longitude pair of the user's location are automatically retrieved from the GPS and the results corresponding to that location are only returned. We use Places API to retrieve the results and then apply re-ranking algorithm. Separate details such as name, contact details, website and address are fetched from each result and displayed in our result page.

3) Re-ranking the Search Results

When a user submits a query on the client, the query containing the user's content and location preferences (i.e., filtered according to the user's privacy setting) are forwarded to the server to obtain the search results from the backend search engine (i.e., Google). The content and location concepts are extracted from the search results and organized .Once the search results are returned by the server, based on whether the search query is content-based or location based, the corresponding databases are referred. The server is used to perform ontology extraction for its speed. The feature vectors from the client are then used in RSVM training to obtain a content weight vector and a location weight vector, representing the user interests based on the user's content and location preferences for the reranking. Again, the training process is performed on the server for its speed. The search results are then reranked according to the weight vectors obtained from the RSVM training. Finally, the reranked results and the extracted ontologies for the personalization of future queries are returned to the client .Re-ranking is performed in two ways: i. Based on highest number of clicks in case of different click values

ii. Displaying top results in case of same number of click values.

i) Based on highest number of clicks in case of different click values:

In this scenario, the retrieved results are compared with the database to see if any clickthroughs are already stored for the query. If there are any clickthroughs stored for that search query, the result that has been visited the most number of times has the highest click value. According to the click values, the results with the maximum clicks are displayed as the top results by sorting them with respect to the clicks in descending order.

ii) Displaying top results in case of same number of click values:

In this scenario too, the retrieved results are compared with the database to see if any clickthroughs are already stored for the query. If there are any click throughs stored for that search query with equal click values for each result, then the results are displayed as top results according to the order in which their clickthroughs are stored in the database.

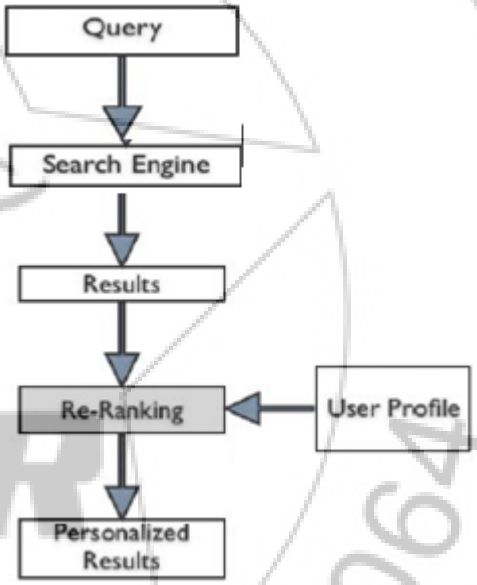

Figure 2: Reranking Process

4) Click through collection and updation:

When the user clicks on a search result, the click through data together with the associated content and location concepts are stored in the click through database on the client. SQLite Database is used for maintaining the database. The clickthroughs are stored on the clients, so the server does not know the exact set of documents that the user has clicked on. Separate databases are maintained for content searches and location-based searches in which the respective clickthrough data is stored.

In addition to this, clickthrough are updated in two ways:

1. If the user clicks on any other link in the re-ranked search results, the click through data of the newly clicked results are also updated in the database which will be helpful for re-ranking the results for any future queries related to the same search.
2. If the user clicks on a result that is already available in the database, the click value of that result is increased.

5) UDD Algorithm

Here, **UDD Algorithm** is used to avoid the duplicate values from final list that will be displayed to the user. So that we may be able to minimize the total number of links as well the duplicate values from the final result page. Unsupervised duplicate Detection (UDD) –Algorithm that uses no predetermined training data to identify duplicates that refer to the same real-world entity. The main aim of UDD is to identify the matching status of each of these records retrieved from multiple web data sources as duplicate and nonduplicate. This is also related to classification problem of each record using only a single class of training example i.e. negative.

UDD consist of following steps:

1. Generation of Similarity Vectors
2. Computation of Potential duplicate vector set P and Non duplicate vector set N
3. Component Weight Allocation
4. Similarity Score calculation
5. Initial Duplicate Identification using WCSS Classifier C1
6. Identifying remaining duplicates from P using SVM Classifier C2
7. Identifying actual duplicate vector set D

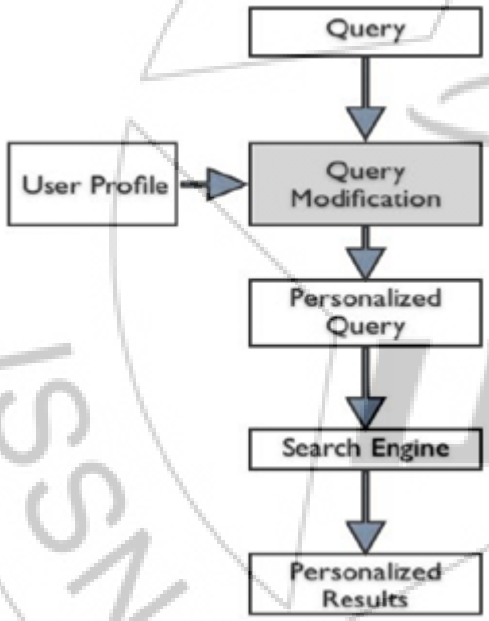

Figure 3: Query Modification Process

Algorithm:

Step 1: Set the parameters W of C1 according to N
Step 2: Use C1 to get a set of duplicate vectors d1 from P and a set of duplicate vectors f from N
Step 3: Remove the identified duplicates from P and N and place into actual duplicate vector set D.
Step 4: Train C2 using D and N'
Step 5: Classify P using C2 and get a set of newly identified duplicate vector pairs
Step 6: Adjust the parameters W of C1 according to N′ and D
Step 7: Use C1 to get a set of duplicate vectors d1′ from P and a set of duplicate vectors f′ from N
Step 8: Return D, repeat the process until no new duplicates are identified by C1.

3.1.2 Ranking SVM

Ranking SVM is an application of Support vector machine, which is used to solve certain ranking problems. The original purpose of Ranking SVM is to improve the performance of the internet search engine. Ranking SVM, one of the pair-wise ranking methods, which is used to adaptively sort the web-pages by their relationships (how relevant) to a specific query. A mapping function is required to define such relationship. The mapping function projects each data pair (inquire and clicked web-page) onto a feature space. These features combined with user's click-through data (which implies page ranks for a specific query) can be considered as the training data for machine learning algorithms.

Generally, Ranking SVM includes three steps in the training period:

1. It maps the similarities between queries and the clicked pages onto certain feature space.
2. It calculates the distances between any two of the vectors obtained in step 1.
3. It forms optimization problem which is similar to SVM classification and solve such problem with the regular SVM solver.

4. Conclusion

Here, the ontology concept is used to group the data as per the related domain. So that, the users can search the most relevant information in specific domain they are requesting, using short queries. The proposed OBPMSE uses content concept and location concepts which are modelled as ontologies. To adapt to the user mobility user's GPS locations are used in the personalization process which helps to improve retrieval effectiveness, especially for location queries. Privacy is addressed by allowing users to control the amount of personal information exposed to the OBPMSE server. UDD Algorithm is used to avoid the duplicate values from final list that will be displayed to the user.

References

[1] E. Agichtein, E. Brill, and S. Domains, "Improving web search ranking by incorporating user behaviour information," in Proc. of ACM SIGIR Conference, 2006, pp.667-686
[2] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning user interaction models for predicting web search result preferences," in Proc. of ACM SIGIR Conference, 2006, pp.321-350 International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 4, April – 2013 ISSN: 2278-0181 www.ijert.org IJERTIJERT 212
[3] G.Y. Chen, T. Suel, and A. Markowetz, "Efficient query processing in geographic web search engines, "in Proc. of ACM SIGIR Conference, 2006, pp.421-462
[4] Q. Gan, J. Attenberg, A. Markowetz, and T. Suel, "Analysis of geographic queries in a search engine log," in Proc. of Loc Web Workshop, 2008, pp.126- 136
[5] T. Joachims, "Optimizing search engines using click through data," in Proc. of ACM SIGKDD Conference, 2002, pp.144-162

[6] K. W. Church, W. Gale, P. Hanks, and D. Hindle, "Using statistics in lexical analysis," in Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon, 1991, pp.531-540
[7] U. W.T. Leung, D. L. Lee, and W.C. Lee,"Personalized web search with location preferences," in Proc. of IEEE ICDE Conference, 2010, pp.208-218
[8] U. Li, Z. Li, W.C. Lee, and D. L. Lee, "A probabilistic topic-based ranking framework for location-sensitive domain information retrieval," in Proc. of ACM SIGIR Conference, 2009.
[9] M. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in Proc.of ICML Conference, 2002.
[10] Bhanupreeti Daggupati, "Unsupervised Duplicate Detection (UDD) Of Query Results from Multiple Web Databases" The Faculty of the Computer Science Program California State University Channel Islands, 2011
[11] Gabriella Pasi "Issues in Personalizing Information Retrieval" IEEE Intelligent Informatics Bulletin,December 2010 Vol.11 No.1
[12] Fang Liu, 1 Clement Yu, 1 Weiyi Meng 2 "Personalized Web Search For Improving Retrieval Effectiveness" University of Illinois at Chicago, Chicago, IL 60607
[13] Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee "PMSE: A Personalized Mobile Search Engine" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL.25, NO. 4, APRIL2013

Author Profile

Talluri. Chakradhar obtained the B.tech. Degree in computer science and engineering from Sri Mittapalli College of engineering, tummalapalem. At present pursuing M.tech in computer science and engineering department at Guntur Engineering College, Guntur.

Miryala Venkatesh obtained B. Tech in Computer science and Engineering in Narasaraopet Engineering College in 2003.MS in latrob university, Australia in 2005, He has Industrial experience for 3Yrs and teaching experience for 7yrs and presently working at Guntur Engineering College, Guntur.