A Novel and Efficient Implementation of User Search Histories

S.N.V.Brahmaji[1] ,N.V.Ashok Kumar[2],Dr.C.Mohan Rao[3].
[1]. *M.Tech II Semester,*[2.] *Assistant Professor,*[3.] *Professor&Principal*
Computer Science & Engineering, Avanthi Institute of Engineering&Technology,AP,India.

Abstract:**Organizing the user search logs is rapidly increasing in the field of data mining for finding the user interestingness and organizing the user search requirements in a proper way .Daily billions of queries can be passed to the server for relevant information, most of the search engines retrieves the information based on the query similarity score or related links with respect to the given query.In this paper we are proposing an efficient clustering mechanism for group up the similar type of query that helps in organizing user search histories.**

I. INTRODUCTION

As of today, the indexed web contains at least 30 billion pages [1]. In fact, the overall web may consist of over 1 trillion unique URLs, more and more of which is being indexed by search engines every day. Out of this morass of data, users typically search for the relevant information that they want by posing search queries to search engines. The problem that the search engines face is that the queries are very diverse and often quite vague and/or ambiguous in terms of user requirements and More questions use similar concept while a single query may correspond to many concepts. We have to order those large data, search engines cluster these queries to group similar items together. To increase usability, most commercial search engines example Google, Yahoo!, Bing, and Ask also augment their search facility through additional services such as query recommendation or query suggestion. These services make it more convenient for users to issue queries and obtain accurate results from the search engine and thus are quite valuable from the search engine perspective and effective clustering of search queries is compulsory pre-requisite for these services to function well. As the size and richness of information storing in the web increasing day by day and so does the variety and the complexity of tasks users try to accomplish online and users cannot navigate with simple queries. Various studies on query logs (e.g., Yahoo's [1] and AltaVista's [2]) reveal that only about 20 percent of queries are navigational and the remaining information is transactional. This happens users expecting more amount of data for future use and managing their finances or planning their purchase decisions.[1]-[3].The primary means of accessing information online is still through keyword queries to a search engine. More Sensitive work is travel arrangement has to be broken down into a number of codependent steps over a period of time. For example user may first search on possible results, timeline, events, etc and After decidingwhen and where user may

then search for the most suitable arrangements for air tickets etc. Each step requires one or more queries and each query results in one or more clicks on relevant pages[5][6].

II.RELATED WORK

The main objective of the project is to retrieve the user interesting results from the search engine based on user query, We introduced a novel approach to solve this problem, with Integrated pattern mining approach. In this approach initially we retrieve the search history which is relevant to user keyword with respect to individual session, On that patterns we apply pattern mining approach to find the optimal patterns then we extract the individual urls from the patterns and displays the optimal results to the user. User searches for required information with known search keyword, search engine receives the query and forwards to the search history. Search history retrieves the session oriented results which are relevant to the search query and forwards these search results to pattern mining approach. Pattern mining approach performs the main task(mining) over the session oriented results, for this approach we are using apriori algorithm for finding the optimal search results. Apriori work on the search oriented results(set of records) and each individual record contains session id and pattern(sequentially navigated urls).Session id indicates the duration between the user initiation and termination. Example search history results as follows, for simplification we are representing the urls in terms of single letters.

The concept of query similarity was originally used in information retrieval studies [2]: measuring the similarity between the content-based keywords of two queries. However, the problem with using this in the query log environment is that users' search interests are not always

the same even if the issued queries contain the same keywords. For instance, the keyword "Apple" may represent a popular kind of fruit whereas it is also the keyword of a popular company "Apple Inc.". Hence, the use of content-based keywords descriptor is rather limited for this purpose. Subsequently, to measure similarity between two queries, the query representation of a vector of URLs in a click through bipartite graph [3][4][5] has been adopted. Nevertheless, no matter how large the query log data set is, it is possible that the complete search intent of some queries may not be adequately represented by the available click-through information. For instance, in a particular large-scale query log, there may be no clicked URL for the query "Honda vs Toyota" and Therefore, even if it is clearly relevant to the query "Honda", on the basis of this click-through data, there is no similarity so that the existing query log data is not accurate for analyzing users' search intent, especially for those queries without any clicked URL. Another reason that causes inaccuracy is that the query log data comprise users' click-through information in a specific period, while search interests might even change over time. If we utilize an aggregated query logs collected in a long period to compare and cluster queries as well as the accuracy may be impacted.

Using Query Keywords :

The first group of related clustering approaches is certainly those that cluster documents containing the keywords. The existing approaches document is represented as a vector in a vector space formed by all the keywords. Researchers have been concerned majorly with the following two conditions:

—similarity function
—algorithms for the clustering process

Using Hyperlinks:

Because of the limitations of key tokens people have been looking for additional criteria for document clustering. Among them one is hyperlinks between documents. The hypothesis is that hyperlinks connect same documents. The idea has been used in some early studies in IR [Garfield 1983] [Kessler 1963]. More recent examples are Google (http://www.google.com) and the authority/hub calculation of Kleinberg [1998]. Although Google does not perform document clustering explicitly, its Page Rank algorithm still results weights of documents. Each document, it is then straightforward to know the documents that are the most strongly related to it according to the weights of the hyperlinks to/from the document. So we can see Page Rank as an implicit clustering approach. Google's use of hyperlinks has been very success and made it one of the best search engines currently available. The same idea is difficult to apply to query clusterin
g and however, because there is no link between queries.

III.PROPOSED WORK

To determine an appropriate clustering method, one first has to choose an appropriate clustering algorithm. So many clustering algorithms available to us and the main characteristics that guide our choice are the following:-As query logs usually are very large and the algorithm should be capable of handling a large data set within reasonable time and space constraints. The algorithm should not require manual setting of the resulting form of the clusters and for example the number or the maximal length of clusters and also unreasonable to determine these parameters in advance.
—Since we only want to find FAQs and the algorithm should filter out those queries with low frequencies.
—Due to the fact that the log data changes the algorithm should be incremental.

DBSCAN does not require the number of clusters as an input parameters and a cluster consists of at least the minimum number of points—MinPts (to eliminate very small clusters as noise); and for every point in the cluster and there exists another point in the same cluster whose distance is less than the distance threshold Eps .This algorithm makes use of a spatial indexing structure (R*-tree) to locate points within the Eps distance from the core points of the clusters. Clusters consists of less than the minimum number of points are considered as "noise" and are discarded. The time complexity of the DBSCAN algorithm is O(n*logn) is average. Previous experiments showed that DBSCAN out performs CLARANS [Ng and Han 1994] by a factor of between 250 and 1900 and increases with the size of the data set. In our approaches it only requires 3 minutes to deal with one-day user logs of 150,000 queries. The efficiency of incremental DBSCAN to update incrementally is due to the density-based nature of the DBSCAN method which is the insertion or deletion of an object only affects the neighborhood of this object. It is based on the formal definition of clusters, it has been proven that the incremental algorithm yields the same results as DBSCAN. The Performance evaluations show Incremental DBSCAN to be more efficient than the basic DBSCAN algorithm

To find a cluster, DBSCAN starts with an arbitrary point p and retrieves all points density reachable from p with respect to Eps and MinPts. p is a core point this procedure yields a cluster with respect to Eps and MinPts. Border Point P has no points are density reachable from p and DBSCAN visits the next point of the database. We use global values for Eps and MinPts and DBSCAN may merge two clusters according to definition 5 into one cluster and if two clusters of different density are "close" to each other. Consider the *distance between two sets of*

points S1 and S2 be defined as dist (S1, S2) = min {dist(p,q) | p∈ S1, q∈ S2}. Then, two sets of points having at least the density of the thinnest cluster will be separated from each other only if the distance between the two sets is larger than Eps. A recursive call of DBSCAN may be necessary for the detected clusters with a higher value for MinPts. However, no disadvantage because the recursive application of DBSCAN yields an elegant and very efficient basic algorithm. The recursive clustering of the points of a cluster is only necessary under conditions that can be easily detected. Below we present a basic version of DBSCAN omitting details of data types and generation of additional information about clusters:

```
DBSCAN (SetOfPoints, Eps, MinPts)
// SetOfPoints is UNCLASSIFIED
ClusterId :=nextId(NOISE);
FOR i FROM 1 TO SetOfPoints.size DO
Point :=SetOfPoints.get(i);
 IF Point.ClId = UNCLASSIFIED THEN
   IF ExpandCluster(Points, Point,
ClusterId, Eps, MinPts) THEN
ClusterId :=nextId(ClusterId)
   END IF
  END IF
 END FOR
END; // DBSCAN
```

SetOfPointSet is either the whole database or a discovered cluster from a previous iteration. Eps and MinPts are the global density parameters determined either manually or according to the heuristics presented. The function PointSet.get(i) returns the i-th element of pointset. The most important function used by DBSCAN is ExpandCluster which is presented below:

```
ExpandCluster(SetOfPoints, Point, ClId, Eps, MinPts) :
Boolean;
seeds:=pointSet.regionQuery(Point,Eps);
 IF seeds.size<MinPts THEN // no core point
SetOfPoint.changeClId(Point,NOISE);
  RETURN False;
ELSE // all points in seeds are density-
    // reachable from Point
PointSet.changeClIds(seeds,ClId);
seeds.delete(Point);
 WHILE seeds <> Empty DO
currentP := seeds.first();
result:= PointSet.regionQuery(currentP,Eps);
IF result.size>= MinPts THEN
   FOR i FROM 1 TO result.size DO
resultP := result.get(i);
   IF resultP.ClId
     IN {UNCLASSIFIED, NOISE} THEN
    IF resultP.ClId = UNCLASSIFIED THEN
seeds.append(resultP);
    END IF;
PointSet.changeClId(resultP,ClId);
   END IF; // UNCLASSIFIED or NOISE
  END FOR;
END IF; // result.size>= MinPts
seeds.delete(currentP);
  END WHILE; // seeds <> Empty
   RETURN True;
  END IF
END; // ExpandCluster
```

A call of PointSet.regionQuery(Point,Eps)returns the Eps-Neighborhood ofPoint in SetOfPoints as a point list. Region queries can be supported efficiently by spatial access methods such as R*-trees
which are assumed to be available in a SDBS for efficient processing of several types of spatial queries (Brinkhoff et al. 1994). The height of an R*-tree is O(log n) for a database of n points in the worst
case and a query with a "small" query region has to traverse only a limited number of paths in the R*-tree. Since the Eps- Neighborhoods are expected to be small compared to the size of the whole data space the average run time complexity of a single region query is O(log n). Each and every point in n points of the database we have at most one region query. So the average run time complexity of DBSCAN is O(n * log n). The ClId (clusterId) of points which have been marked to be NOISE may be changed later if they are density-reachable from some other point of the database and happens for border points of a cluster and These points are not added to the seeds-list because we already know that a point with a ClId of NOISE is not a core point and Adding those points to seeds would only result in additional region queries which would yield. If two clusters C1 and C2 are very close to each other and it might happen that some point p belongs to both C1 and C2. The p must be a border point in both clusters because otherwise C1 would be equal to C2 since we use global parameters. Here point p will be assigned to the cluster discovered first except from these rare situations and the result of DBSCAN is independent of the order in which the points of the database are visited due to Lemma 2.

In approach of how to determine initial Parameters *Eps*and *MinPts*is to look at the behavior of the distance from a point to its **k**th nearest neighbor which is called **k**-dist and the **k**-dists are computed for all the data points for some k,sorted in ascending order, and then plotted using the sorted values, as a result and a sharp change is expected. The sharp change at the value of **k**-dist corresponds to a suitable value of *Eps*. Note that the value of *Eps*that is determined in this way depends on **k** and but does not change dramatically as **k** changes. Because DBSCAN uses a density-based definition of a cluster and it is relatively resistant to noise and can handle clusters of different shapes and sizes. DBSCAN can find many clusters that could not be found using some other clustering algorithms. However, the main weakness of DBSCAN is that it has trouble when the clusters have greatly varied densities. To reduce over the limitations of DBSCAN and VDBSCAN is acquainted. Firstly, VDBSCAN calculates

and stores k-dist for each project and partition k-dist plots. Next the number of densities is given intuitively by k-dist plot. Thirdly, choose parameters *Epsi*automatically for each density. Finally scan the dataset and cluster different densities using corresponding *Epsi*. And finally, display the valid clusters corresponding with varied densities.

IV.CONCLUSION

In this paper we enhanced the mechanism of organizing the user search histories by providing the improved dbscan algorithm, it removes the unnecessary data points (Query url links).It is variable length and we need not to specify the number of clusters prior clustering. In this improved dbscan algorithm density factor is depends on k-dist plot. Here that generates the optimal clusters

REFERENCES

[1] "http://www.worldwidewebsize.com/."
[2] G. Salton and M. J. McGill, *Introduction to Modern InformationRetrieval*. New York, NY, USA: McGraw-Hill, Inc.,1986.
[3] D. Beeferman and A. L. Berger, "Agglomerative clusteringof a search engine query log," in *KDD*, pp. 407–416, 2000.
[4] J.-R. Wen, J.-Y.Nie, and H. Zhang, "Query clustering usinguser logs," *ACM Trans. Inf. Syst.*, vol. 20, no. 1, pp. 59–81,2002.
[5] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li,
"Context-aware query suggestion by mining click-through and session data," in *KDD*, pp. 875–883, ACM, 2008.
[6] T. Joachims, "Optimizing search engines usingclickthroughdata," in *KDD*, pp. 133–142, ACM, 2002.
[7] E. Agichtein, E. Brill, and S. T. Dumais, "Improving websearch ranking by incorporating user behavior information,"
in*SIGIR*, pp. 19–26, 2006.
[8] U. Irmak, V. von Brzeski, and R. Kraft, "Contextual rankingof keywords using click data," in *ICDE*, pp. 457–468, 2009.
[9] F. Radlinski and T. Joachims, "Query chains: learning to rankfrom implicit feedback," in *KDD*, pp. 239–248, 2005.
[10] T. Joachims, L. A. Granka, B. Pan, H. Hembrooke, F. Radlinski,and G. Gay, "Evaluating the accuracy of implicit feedbackfrom clicks and query reformulations in web search," *ACMTrans. Inf. Syst.*, vol. 25, no. 2, 2007.
[11] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top klists," *SIAM J. Discrete Math.*, vol. 17, no. 1, pp. 134–160,2003.
[12] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization:algorithms and complexity*. Upper Saddle River, NJ,USA: Prentice-Hall, Inc., 1982.
[13] M. Barbaro and T. Z. Jr., "A face is exposed for aol searcherno. 4417749," Augest 9, 2006.(New York Times).
[14] K. Hafner, "Researchers yearn to use aol logs, but they hesitate," Augest 23, 2006. (New York Times).
[15] H. Cui, J.-R. Wen, J.-Y.Nie, and W.-Y. Ma, "Query expansionby mining user logs," *IEEE Trans. Knowl. Data Eng.*, vol. 15,
no. 4, pp. 829–839, 2003.
[16] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Query recommendationusing query logs in search engines," in *EDBT*,2004.
[17] J.-R. Wen, J.-Y.Nie, and H.-J. Zhang, "Clustering userqueries of a search engine," in *WWW '01*.
[18] M. Ester, H.-P.Kriegel, J. Sander, and X. Xu, "A densitybasedalgorithm for discovering clusters in large spatialdatabases with noise," in *KDD*, pp. 226–231, 1996.
[19] E. Yilmaz, J. A. Aslam, and S. Robertson, "A new rankcorrelation coefficient for information retrieval," in *SIGIR*,pp. 587–594, 2008.
[20] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on In Pattern Analysis and Machine Intelligence*, vol. PAMI-1, pp. 224–227, Nov. 1977.
[21] S. Saitta, B. Raphael, and I. F. C. Smith, "A bounded indexfor cluster validity," in *MLDM*, pp. 174–187, 2007.

AUTHORS DETAILS::

S.N.V.Brahmaji received the B.Tech degree in Computer Science and Engineering from SRKR Engineering College, Bhimavaram, A.P in 2008. He is pursuing his M.Tech in Computer Science and Engineering in Avanthi Institute of Engineering &Technology,Vizag,A.P.His research interests include RDBMS, Data Warehousing and Data Mining, and Secure Database Applications.

N.V.AshokKumar.M.Tech (CSE)
He received the B.Tech degree in Computer Science and Engineering from JNT University, Kukatpalli, Hyderabad and received the M.Tech degree in Computer Science and Technology from JNT University, Kakinada. Presently he is working as Assistant Professor in Computer Science and Engineering in Avanthi Institute of Engineering and Technology,Vizag, A.P.His research interests include Network Security, Data Warehousing and Data Mining and RDBMS .He has Published more than 10 papers in various national and international journals.

Dr.C.MohanRao.M.Tech (CSE), Ph.D.
He received the M.Tech degree in Computer Science and Technology from Andhra University College of Engineering,Vizag and awarded PhD by Andhra University in 2000.He has 18 years of teaching and research experience and guided number of M.Tech students for their projects.Presently he is working as Principal in

Avanthi Institute of Engineering and Technology, Vizag,A.P.His research interests include Data Warehousing and DataMining,Cryptography and Network Security and Artificial Intelligence..He has published 23 papers in various national and international journals.He is guiding 2 research scholars for Ph.D.He received the Best Teacher Award from JNTU,Kakinada in 2009.

