International Journal of Advanced Computer Technology (IJACT)

ISSN:2319-7900

USER SPECIFIC SEARCH HISTORIES AND ORGANIZING PROBLEMS

M.Anusha*[1]
M.Tech *Department of Computer Science*
Kakatiya Institute of Technology and Science,
Warangal

Dr.P.Niranjan*[2]
professor, Department of computer science,
Kakatiya Institute of technology and science,
Warangal

Dr. P.Shireesha *[3]
Asst.professor, Department of Computer Science and Engineering,
Kakatiya Institute of Technology and Science
Warangal

Abstract

Creating search histories are a difficult process in the web. The user search logs are rapidly increasing in the field of data mining for finding the user interestingness. Present days more number of queries can be passed to the server for relevant information most of the search engines retrieves the information based on the query similarity related links with respect to the given query. This paper we stressed on the concept explains the problem of organizing a user's historical queries into groups in a dynamic and automated fashion. 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.

Introduction

Search mechanism is the keyword explaining the way we store and retrieve data at perfect appropriation and based on the requirement. The relationships between the searched and searching data the reformulated queries come into existence. We change this model to hierarchical query model distribution. As of today the indexed web contains at least 30 billion pages. 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 users typically search for the relevant information that they want by posing search queries to search engines. The problem is that the queries are very diverse and often quite vague and or ambiguous in terms of user basic inputs. Most of the individual queries may refer to a single concept, while a single query may correspond to several techniques. For organize and bring some order to this massive unstructured dataset search engines cluster these queries together to group similar items. To increase usability, most commercial search engines and also augments 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 web search engine, and thus it is quite valuable. From the search engine view efficient group of search queries is a necessary pre requisite for these services to function well. As the size and richness of information increases on web because does the variety and the complexity of tasks the users try to complete online. End users are no longer content with issuing simple navigational queries. The remaining is informational or transactional. Since users now pursue much broader informational and task oriented goals such as arranging for travel of future, managing their finances or planning their purchase plans. However the primary means of accessing information online is still through keyword queries to a web search engine. Each step requires one or more queries, and each query results clicks on relevant pages. The K-means algorithm is one of the most frequently used investigatory algorithms in data analysis. The proposed approach of algorithm finds to locate K models or averages throughout a data set in such a way that the K prototypes in some way best represents the data. However the algorithm is known to suffer from the defect that the means or prototypes found depend on the initial values given to them at the start of the simulation: a typical program will converge to a local optimum. There are a number of heuristics in the literature which attempt to address this issue but the fault lies in the performance function on which K-means is based. In this paper we are introducing an enhanced dbscan algorithm for organizing the user search histories.

Related Work

In order to make the things proper, we need to explore things which make us to explore. In this paper we explore and evaluate strategies for how to automatically generate for learning retrieval of functions from observed user behavior on the approach of used data or clicked data. The concept of

ISSN:2319-7900

query similarity was originally used in information retrieval studies: 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.". Subsequently to measure similarity between two queries, the query representation of a vector of URLs in a click through bipartite graph has been adopted. No matter how large the query log data set it is possible that the complete search intent of some queries may not be adequately represented by the available click-through information. In a large scale query log, there may be no clicked URL for the query Honda vs. Toyota. But there is no relevant to the query Honda on the basis of this click through data there is no similarity. Therefore existing query log data is not accurate enough for analyzing users' search intent especially for those queries without clicked any URL.

Using Query Keywords

The first group of related clustering approaches is certainly those that cluster documents using the keywords it. In proposed approaches in general a document is represented as a vector in a vector space generated by the keywords. Academicians have been concerned mostly with the following two aspects:

a) Similarity function.

b) Algorithms for the clustering process.

Keyword based document clustering has provided interesting results. One contributing factor is the large number of keywords contained in documents. Even if some of the keywords of two similar documents are different, there are still many others that can make the documents similar in the similarity calculation. However because, specifically the queries submitted to the web search engines usually are very short in many cases it is hard to deduce the semantics from the query itself. Therefore keyword alone does not provide a reliable basis for clustering queries effectively. In addition, words such as where and who are treated as stop words in traditional IR methods. For questions however these words encode important information about the user's requirement specifically in the new-generation web search engines such as Ask Jeeves. Considering an example whose input parameter is the user intends to find information about a person. So even if a keyword based approach is used in query clustering, it should be modified from that used in traditional clustering document. The entire question is represented as a template in accordance with the question type. During input question evaluation, the input requirement template may be elaborated using a morphological changes, in our proposed case, we found that well-formed natural language input questions represented only a small part of queries. Most probable queries are simply short parts of phrases or keywords.

Using Hyperlinks

Because of the limitations of keywords people have been looking for additional criteria for document clustering or grouping. One of them is the hyperlinks between groups of documents. The hypothesis is related hyperlinks connect similar documents. The proposed idea had been used in some early studies in IR. More recent examples are Google and the hub calculation of Kleinberg. Although Google does not perform document clustering explicitly, its Page Rank algorithm still results in a weighting of hyperlinks to a document, it is then direct to know the documents that are the most strongly related to it according to the weights of the hyperlinks from the document. Google's use of hyperlinks has been mostly successful, organizing and making it one of the best search engines currently available. The same idea is difficult to apply to query clustering because there is no link between Queries.

Proposed Work

To determine an appropriate clustering has to choose an appropriate clustering algorithm. There are many clustering algorithms available. The main characteristics that guide are the following: As query logs usually are more, the algorithm approach 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 for consideration number of maximal size of final clusters. It is not reasonable to determine these Parameters in advance. Because we only need to find frequently asked questions, the proposed algorithm should filter out those queries with low frequencies. Due to that reasons the log data changes daily, the algorithm should be incremental. The density-based clustering method DB SCAN and its incremental version Incremental DB SCAN satisfy the above requirements. Our proposed DB SCAN does not require the number of clusters as an input parameters the cluster consists of at least the min number of points—MinPts (to eliminate very small clusters as noise); and for each point in the cluster and there is another point in the same cluster whose distance is less than the distance threshold Eps. This approach makes use of a spatial indexing structure (R*-tree) to locate points within the Eps distance from the core points of the clusters. Total clusters consisting of less than the minimum number of points are considered as "noise" and had been discarded. Average time complexity of this DB SCAN algorithm is O (n*logn). Previous experiments showed that DB SCAN outperforms CLARANS by a factor of between 250 and 1900;

ISSN:2319-7900

it increases with the size of the data set. Its ability of Incremental DB SCAN to update incrementally is due to the density-based nature of the DB SCAN approach , which the insertion or deletion of an object only affects the neighborhood of this entity and based on the formal definition of clusters, it has been proven that the incremental algorithm yields the same results as DB SCAN. To find a cluster, DB SCAN starts with an arbitrary point p and retrieves all points density-reachable from p with respect to. Min distance (Eps) and minimum number of points (Min Pts). If p is a core point, it procedure produces a cluster. If p consider is a border point p and no points are density- reachable from p and DB SCAN visits the next point of the database therefore use global values for Eps and MinPts ,the algorithm DB SCAN may merge two clusters according to definition 5 into single cluster, if two clusters of different density are "close" to each other. Let us 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 min scale distance and consequently. There is no disadvantage because the recursive application of DB SCAN yields an elegant and very efficient basic approach. Furthermore and recursive clustering of the points of a cluster is only necessary under conditions that can be easily detected and the following, we present a basic version of DB SCAN omitting details of data types and generation of additional information about clusters:

```
DB SCAN (Set of Points, Eps, MinPts)
// Set of Points is UNCLASSIFIED
Clustered: =   next Id (NOISE);
FOR i FROM 1 TO Set of Points .size DO
Point =   Set Of Points. Get (i);
IF Point. Cl Id = UNCLASSIFIED THEN
  IF Expand Cluster (Set Of Points, Point,
    Cluster Id, Eps, MinPts) THEN
    Cluster Id = next Id (Cluster Id)
  END IF
 END IF
END FOR
END; // DBSCAN
```

The function Set_Of_Points.get (i) returns the i-th element of Set_Of_Points and most important function used by DB SCAN is Expand Cluster Point in Set_Of_Points as a list of points. 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. 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 complexity and average run time complexity of a single region query as O (log n) is defined for each of the n points of the data, we had at most one region query and Thus average run time complexity of DB SCAN is as O (n* log n). The cluster Id of points which have been marked to be NOISE may be changed later, when they are density-reachable from some other point of the data and This 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 Cl Id of NOISE is not a core point and by Adding those points to seeds would only result in additional region queries which would yield no generated answers. when two clusters C1 and C2 are very close to each other, it may occurs that some point p belongs to both C1 and C2 and then p must be a border point in both clusters because otherwise C1 would be equal to C2 since we use global constants, In this case, point p will be assigned to the cluster discovered initially and excluding from these rare situations, the result of DB SCAN is Independent of the order in which the points of the database are visited due to Lemma 2. The basic approach of how to determine the parameters Eps and MinPts is to look at the behavior of the distance from a point to its kth nearest neighbor and which is called k-dist and these k-distance are computed for all the data points for some number of k, points sorted in ascending order and then plotted using the sorted values as a result leads to a sharp change is expected to see. The sharp change at the value of k dist corresponds to a suitable value. Note that the value of Eps that is determined in this way depends on k, but does not change dramatically as k changes. Because DB SCAN uses a density-based definition of a cluster and it is relatively resistant to noise and can handle clusters of different shapes and sizes. Thus, DB SCAN can find many clusters that could not be found using some other clustering algorithms like K-means, always the main weakness of DB SCAN is that it has trouble when the clusters have greatly density varies to sweep over the limitations of DB SCAN. Firstly DB SCAN calculates and stores k-dist for each project & partition k-Dist plots. Secondly the number of densities is given intuitively by k-dist plot. Thirdly, choose parameters automatically for each density.

Incremental Algorithm

Incremental algorithms are radically different from static methods for the way they build and use Recommendation models. While static algorithms need an off-line preprocessing phase to build the model from scratch every time an update of the knowledge base is needed, incremental algorithms consist of a single online module integrating the two functionalities:

i) Updating the model.

ii) Providing suggestions for each query.

The two incremental algorithms differ from their static counterparts by the way in which they use data and manage to build the model. Both algorithms exploit LRU caches and Hash tables to store and retrieve efficiently queries and links during the model update. Our two incremental algorithms are inspired by the Data Stream Model in which streams of queries are processed by database system. Queries consists modification of values associated with a set of data. An algorithm in the data stream model must decide at each time step which subset of the set of data is worthwhile to maintain in memory. The goal is to attain an approximation of the results we would have had in the case of the non- streaming model. Make a first step towards a data stream model algorithmic framework aimed at building query recommendations. The first uses association rules while the second exploits click-through data. Below Fig.1 explains entire work of this paper. User first enters the query for getting efficient results. The search engine compares the entered query with existing query log.

If it is existed in the query log, the search engine applies incremental algorithm for that entry and provides results to user. The incremental algorithm includes I Association rule and I Cover graph.

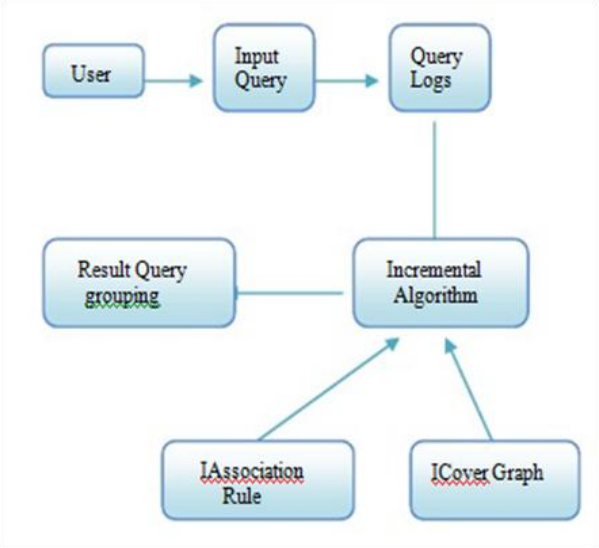

Figure 1. Architectural Daigram

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. 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.

We propose a novel reformulation framework that transforms the original query into a distribution of reformulated queries where each reformulated query is associated with a probability indicating for retrieval. The query distribution model considers a reformulated query as the basic unit thus explicitly modeling how query concepts are used together to form a realistic or actual query. The first two aspects can be efficiently implemented when large scale query logs are available. We can limit the reformulated queries to those appearing in query logs. In this way instead of generating queries we can simply search the query logs, which can be efficiently implemented using the index. Also, which speed up the query feature extraction for the retrieval aspect, instead of running multiple reformulated queries, we reuse the retrieval scores of the words and phrases shared by these queries.

References

[1] Goldberg, J., Stimson, M., Lowenstein, M., Scott, M., and Wichansky, A. 2002. Eye tracking in web search tasks: design implications. In Proceedings of the Eye tracking Research and Applications Symposium (ETRA).

[2] G. Salton and M. J. McGill, Introduction to Modern Information Retrieval. New York, NY, USA: McGraw-Hill, Inc., 1986.

[3] D. Beeferman and A. L. Berger, “Agglomerative clustering of a search engine query log,” in KDD, pp. 407– 416, 2000.

[4] J.-R. Wen J.Y. Nie, and H. Zhang, “Query clustering using user 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,N 2008.

[6] T. Joachim, “Optimizing search engines using click through data,” in KDD, pp. 133–142, ACM, 2002.

[7] E. Agichtein, E Brill, and S. T.Dumais, “Improving web search ranking by incorporating user behavior information,” in SIGIR, pp. 19–26, 2006.

[8] U. Irmak, V. von Brzeski, and R. Kraft, “Contextual ranking of keywords using click data,” in ICDE, pp. 457– 468, 2009.

[9] F. Radlinski and T. Joachim, “Query chains: learning to rank from implicit feedback,” in KDD, pp. 239–248, 2005.

[10] T. Joachim, L. A. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay, “Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search,” ACM Trans. Inf. Syst., vol. 25, no. 2, 2007.

ISSN:2319-7900

Biographies

M. ANUSHA is currently pursuing her M.Tech. Computer Science & Engineering in Kakatiya Institute of Technology and Science, Warangal. She received her B.Tech in Computer Science and Engineering from Balaji Institute of Technology and Science, Narsampet, Warangal. Her area of interests includes Data mining and Software Engineering.

DR. NIRANJAN POLALA is working as Professor and HOD of CSE in KITS, Warangal. He received Ph.D in CSE from Kakatiya University, Warangal in the year 2013. He received M.Tech (Computer Science and Engineering) from NIT, Warangal in the year 2001 and B.E Computer Science from Nagpur University in 1992. He authored three text books in the field of computer science. He published 30 research papers in various International Journals and Conferences. He is a member of the ISTE and CSI. His area of interests includes Software Engineering.

DR. SHIREESHA PAKALA is working as Assistant Professor in Department of CSE, KITS, Warangal. She received Ph.D in Computer Science from Kakatiya University, Warangal in the year 2012. She received M.Sc. Computer Science from Kakatiya University in 2001. She published 8 research papers in various International Journals and International Conference. She is the member of the ISTE and IETE. Her area of interests includes Data mining and Software Engineering.