International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-2 Issue-4, March 2013

An Efficient Personalized Web Search Mechanism using BinRank Algorithm

T.D. Khadtare, P.R. Thakare, S.A.J. Patel

Abstract*— Dynamic authority-based online keyword search algorithms, such as Object rank and personalized page rank leverage semantic link information to provide high quality, high recall search in databases and the web. Conceptually, these algorithms require a querytime page rank style iterative computation over the full graph. This computation is too expensive for large graphs and not feasible at query time. Alternatively, building an index of precomputed results for some or all keywords involves very expensive processing. We introduce BinRank, a system that approximates ObjectRank results by utilizing a hybrid approach inspired by materialized views in traditional query processing. We materialized relatively small subsets of the data graph so that any keyword query can be answered by running ObjectRank on only one of the subgraphs. BinRank generates the subgraph by partitioning all the terms in corpus based on their cooccurence, executing ObjectRank for each partition using the terms to generate a set of random walk starting points, and keeping only those objects that receive negligible score. We demonstrate that Binrank can achieve subsecond query execution time on the English Wikipedia data set, while producing high-quality search results that closely approximate the results of ObjectRank on the original graph. Our experimental evaluation investigates the trade-off between query execution time, quality of results, and storage requirements of BinRank.

Keywords*— BinRank, ObjectRank, Online keyword search, Page Rank .

I. INTRODUCTION

Existing two keyword search algorithms, such as ObjectRank and personalized PageRank provides high quality and high recall search in databases, and the web. These algorithms require a querytime for iterative computation over the full graph which is too expensive for large graph, and not feasible at query time. Page Rank algorithm utilizes the Web graph link structure to assign global importance to Web pages.[1][2]. The Page Rank score is independent of a keyword query. Recently, dynamic versions of the PageRank algorithm have become popular. They are characterized by a query-specific choice of the random walk starting points. Personalized Page Rank (PPR) is used for Web graph data sets. PPR performs search personalized on a preference set that contains web pages that a user likes. For a given preference set, PPR performs a very expensive fixpoint iterative computation over the entire web graph, while it generates personalized search results. Therefore, the issue of scalability of PPR has attracted a lot of attention. Page Rank does not use Keyword search method to extract useful information. ObjectRank overcomes the drawback of PageRank. It extends PPR to perform keyword search in databases. ObjectRank uses a query term posting list as a set of random walk starting points and conducts the walk on the instance graph of the database[5]. However, ObjectRank suffers from the same scalability issues as personalized PageRank, as it requires multiple iterations over all nodes and links of the entire database graph. Here, we introduce a system using BinRank that employs a hybrid approach where query time can be traded off for preprocessing time and storage[7]. BinRank closely approximates ObjectRank scores by running the same ObjectRank algorithm on a small subgraph instead of the full data graph. Subgraphs are precomputed offline. The precomputation can be parallelized with linear scalability. BinRank query execution easily scales to large cluster by distributing the subgraphs between the nodes of cluster. This way more subgraphs can be kept in RAM, thus decreasing the average query execution time. Since the distribution of the query terms in a dictionary is usually very uneven, the throughput of the system is greatly improved by keeping duplicates of popular subgraphs on multiple nodes of the cluster. The Query term is routed to the least busy node that has the corresponding subgraph. Thus, BinRank algorithm finds application in efficient web mining and database searching.

II. CONVENTIONAL ALGORITHMS

The issue of scalability of PPR has attracted a lot of attention. PPR performs a very expensive fixpoint iterative computation over the entire graph, while it generates personalized search results[3]. To avoid the expensive iterative calculation at runtime, one can honestly precompute and materialize all the possible personalized PageRank vectors (PPVs). Although this method guarantees fast user response time, such precomputation is impractical as it requires a huge amount of time and storage especially when done on large graphs. In this section, we examine hub-based and Monte Carlo style methods that address the scalability problem of PPR, and give an overview of HubRank that integrates the two approaches to improve the scalability of ObjectRank. Even though these approaches enabled PPR to be executed on large graphs, they either limit the degree of personalization or deteriorate the quality of the top-k result lists significantly. Hub-based approaches materialize only a selected subset of PPVs.

Manuscript published on 30 March 2013.
*Correspondence Author(s)
Prof. T.D. Khadtare, Department of Information Technology, Sinhgad Institute of Technology and Science, University of Pune, Pune, India.
Prof. P.R. Thakare, Department of Information Technology, Sinhgad Institute of Technology and Science, University of Pune, Pune, India.
Prof. S.A.J. Patel, Department of Information Technology, Sinhgad Institute of Technology and Science, University of Pune, Pune, India.

Topic-sensitive PageRank suggests materialization of 16 PPVs of selected topics and linearly combining them at query time. The personalized PageRank computation suggested in enables a finer-grained personalization by efficiently materializing significantly more PPVs (e.g., 100 K) and combining them using the hub decomposition theorem and dynamic programming techniques. However, it is still not a fully personalized PageRank, because it can personalize only on a preference set subsumed within a hub set H. Monte Carlo methods replace the expensive power iteration algorithm with a randomized approximation algorithm. In order to personalize PageRank on any arbitrary preference set with maintaining just a small amount of precomputed results, Fogaras et al. introduce the fingerprint algorithm that simulates the random walk model of PageRank and stored the ending nodes of sampled walks[3]. Since each random walk is independent, fingerprint generation can be easily parallelized and the quality of search results improves as the number of fingerprints increases. However, the precision of search results generated by the fingerprint algorithm is somewhat less than that of power-iteration-based algorithms, and sometimes, the quality of its results may be inadequate especially for nodes that have many close neighbors. Monte Carlo algorithm takes into account not only the last visited nodes, but also all visited nodes during the sampled walks. Also, it showed that Monte Carlo algorithms with iterative start outperform those with random start. Another approach is HubRank which is a search system based on ObjectRank that improved the scalability of ObjectRank by combining the above two approaches. It first selects a fixed number of hub nodes by using a greedy hub selection algorithm that utilizes a query workload in order to minimize the query execution time. Given a set of hub nodes H, it materializes the fingerprints of hub nodes in H. At query time, it generates an active subgraph by expanding the base set with its neighbors. It stops following a path when it encounters a hub node whose PPV was materialized, or the distance from the base set exceeds a fixed maximum length [7]. HubRank recursively approximates PPVs of all active nodes, terminating with computation of PPV for the query node itself. During this computation, the PPV approximations are dynamically pruned in order to keep them sparse. Dynamic pruning takes a key role in outperforming ObjectRank by a noticeable margin. However, by limiting the precision of hub vectors, HubRank may get somewhat inaccurate search results, as stated in. Also, since it materialized only PPVs of H, just as, the efficiency of query processing and the quality of query results are very sensitive to the size of H and the hub selection scheme. BinRank can strike a good balance between query performance and closeness of approximation.

III. PROPOSED METHODOLOGY

BinRank Algorithm

We propose a BinRank system that provides user with personalized search and reduce query processing time and storage. For this, it employs a hybrid approach. This system maintains a separate record for each user and stores his/her search results according to the relevance of documents. User can maintain his/her personal search catalogue, can see his/her own search history, can get the results for queries which are stored in database within less time than time required to it search over web. It provides a fast search over previous queries giving the user an optimized result. User can select the category words for searching and he can add his own category words. This system also maintains ranking list of frequently used websites. The algorithm used for Bin Construction is given below.

Input: A set of workload terms W, with their posting lists
Output: A set of bins B
While W is not empty **do**
Create a new empty bin b
Create an empty cache of candidate terms C
Pick term t Є W with the largest posting list size |t|
While t is not null **do**
Add t to b, and remove it from W
Compute a set of terms T that co-occur with t
For each t' Є T **do**
Insert (or update) mapping <t', null> into C
End for each
bestI:=0
for each mapping <c, i> Є C **do**
if i= null **then** // b ∩ c has not been computed yet
i:=| b∩c|
update mapping <c, i> in C

End if
Union:= |b| + |c| - i
If *union > maxBinSize* ***then***
Remove <c, i> from C
Else if *i > bestI* ***then***
bestI:= i
t:=c
end if
end for each
if bestI=0 **then** // no candidate left
pick t Є W with maximum |t| ≤ maxBinSize - |b|
if no such t exists , t:=null
end if
end while
add completed b to B
end while

IV. SYSTEM ARCHITECTURE

Fig.1 shows the architecture of Bin Rank System. During the preprocessing stage (left side of figure) we generate MSG's(materialized subgraph). During query processing stage (right side of figure) we execute the ObjectRank algorithm on the subgraphs instead of full graph and produce high quality approximations of top-k lists at a small cost. In order to save preprocessing cost and storage, each MSG is designed to answer multiple term queries. System architecture is divided in two stages.

- Preprocessing
- Query Processing

A. Preprocessing

The preprocessing stage of BinRank starts with a set of workload terms W for which MSGs will be materialized. If an actual query workload is not available, W includes the entire set of terms found in the corpus. We exclude from W all terms with posting lists longer than a system parameter maxPostingList.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The posting lists of these terms are deemed too large to be packed into bins. We execute ObjectRank for each such term individually and store the resulting top-k lists. Naturally, maxPostingList should be tuned so that there are relatively few of these frequent terms. For each term w € W, BinRank reads a posting list T from the Lucene3 index and creates a KMV synopsis T that is used to estimate set intersections. The bin construction algorithm, PackTermsIntoBins, partitions W into a set of bins composed of frequently co-occurring terms. The algorithm takes a single parameter maxBinSize, which limits the size of a bin posting list, i.e. the union of posting lists of all terms in the bin. During the bin construction, BinRank stores the bin identifier of each term into the Lucene index as an additional field. This allows us to map each term to the corresponding bin and MSG at query time. The ObjectRank module takes as input a set of bin posting lists B and the entire graph G(V,E) with a set of ObjectRank parameters, the damping factor d, and the threshold value €. The threshold determines the convergence of the algorithm as well as the minimum ObjectRank score of MSG nodes. Our ObjectRank implementation stores a graph as a row compressed adjacency matrix. In case that the entire data graph does not fit in main memory, we can apply parallel PageRank computation techniques such as hypergraph partitioning schemes. The MSG generator takes the graph G and the ObjectRank result with respect to a term bin b, and then, constructs a subgraph Gb(V',E') by including only node with rt (u)≥€b. €b is the convergence threshold of b, that is, €/|BS(b)|. Given the set of MSG nodes V', the corresponding set of edges E' is copied from the in-memory copy of G. The edge construction takes 1.5-2 seconds for a typical MSG with about 5 million edges. Once the MSG is constructed in memory, it is serialized to a binary file on disk in the same row-compressed adjacency matrix format to facilitate fast deserialization.

B. Query Processing

For a given keyword query q, the query dispatcher retrieves from the Lucene index the posting list bs(q) (used as the base set for the ObjectRank execution) and the bin identifier b(q). Given a bin identifier, the MSG mapper determines whether the corresponding MSG is already in memory. If it is not, the MSG deserializer reads the MSG representation from disk. The BinRank query processing module uses all available memory as an LRU cache of MSGs. For smaller data graphs, it is possible to dramatically reduce MSG storage requirements by storing only a set of MSGnodes V', and generating the corresponding set of edges E' only at query time. However, in our Wikipedia, data set that would introduce an additional delay of 1.5-2 seconds, which is not acceptable in a keyword search system. The ObjectRank module gets the in-memory instance of MSG, the base set, and a set of ObjectRank calibrating parameters: 1) the damping factor d; 2) the convergence threshold €; and 3) the number of top-k list entries k. Once the ObjectRank scores are computed and sorted, the resulting document ids are used to retrieve and present the top-k objects to the user. Multikeyword queries are processed as follows:

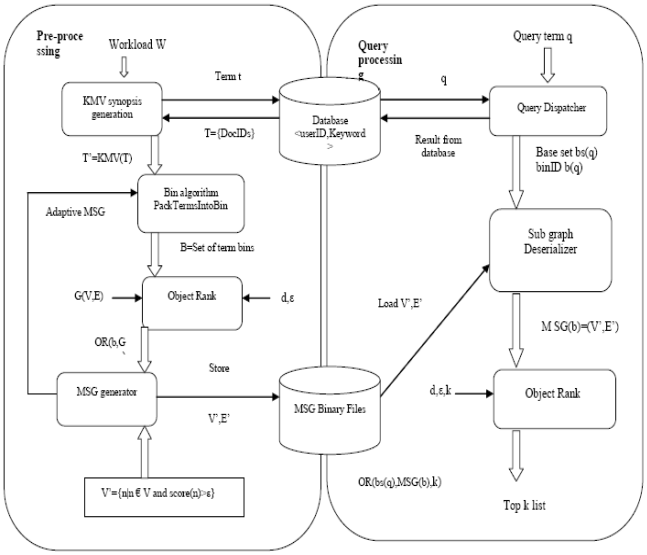


Fig.1. Architecture of BinRank system

For a given conjunctive query composed of n terms {t1,…, tn}, the ObjectRank module gets MSGs, {MSG(b(t1)),…MSG(b(tn))} and evaluates each term over the corresponding MSG. Then, it multiplies the ObjectRank scores obtained over MSGs to generate the top-k list for the query. For a disjunctive query, the ObjectRank module sums the ObjectRank scores with respect to each term calculated using MSGs to produce BinRank scores. One of the advantages of BinRank query execution engine is that it can easily utilize large clusters of nodes.

V. SYSTEM COMPONENTS

A. User Registration

We are providing the facility to register new users. If anyone wants use to this application, they should become a member of this application. To get the membership login, users should make registration with application. In registration system will get all the details about the users and it will be stored in a database to create membership.

B. Authentication

This module provides the authentication to the users who are using application. In this module we are providing the registration for new users and login for existing users.

C. Search Query Submission

Users query will be submitted in this module. Users can search for any kind of information application when we connect with Internet. Users query will be processed based on their submission, and then it will produce the appropriate result. Result will be produced based on our algorithm.

D. Index Creation

Index is something like the count of search and result which we produced while searching. Based on the index we will create the rank for the results, such like pages or corresponding websites. This will be maintained in background for future use like cache memory. We are creating the index to speed up the search. Search is efficient and fast with the help of implementing BinRank algorithm.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

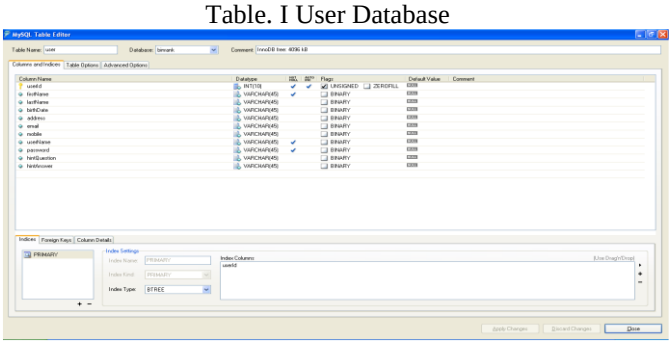
E. Query Processing and Graph Generation

We generate a subgraph that contains all objects and links relevant to a set of related terms and have all the information needed to rank objects. Based on the index creation we need to generate the results for the users query. BinRank algorithm will use the indexing and ranking techniques to produce the efficient results in short time.

VI. RESULTS AND DISCUSSIONS

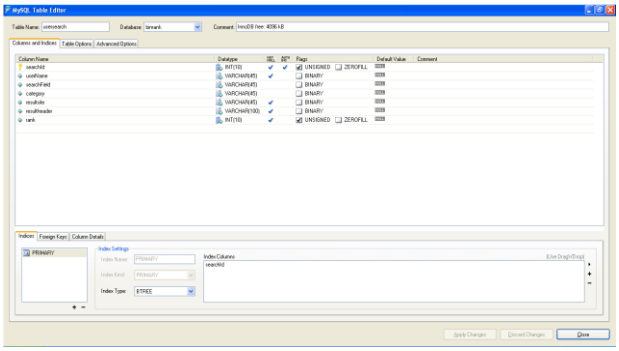
We used MY SQL for creating Database. We required three tables for BinRank Database.
These are as follows-

Table. I User Database



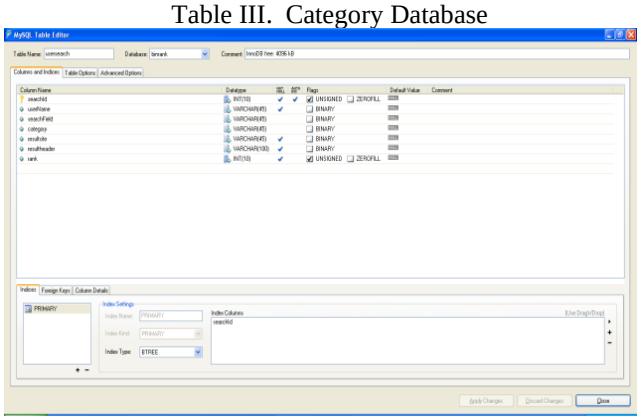
We created this table to store the user details who registered to BinRank

Table II. User Search Database

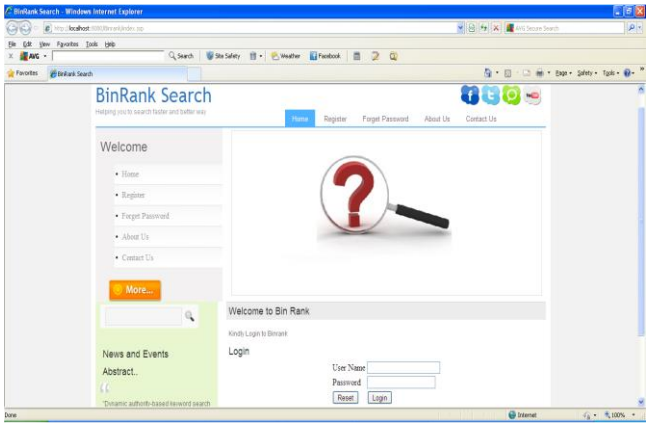


This table is required to store all the information regarding particular user's search.

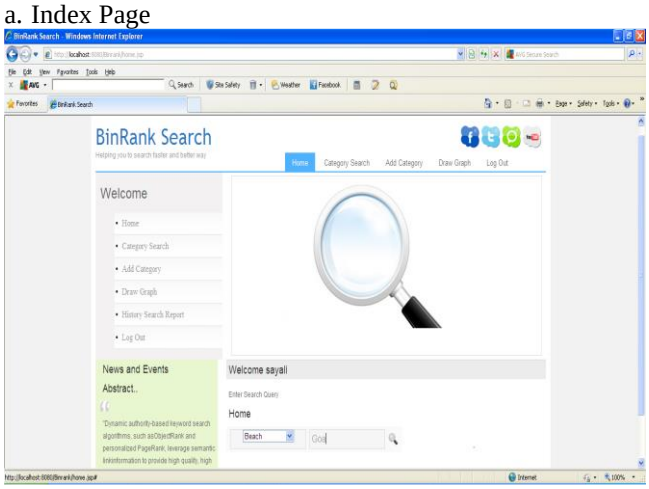
Table III. Category Database



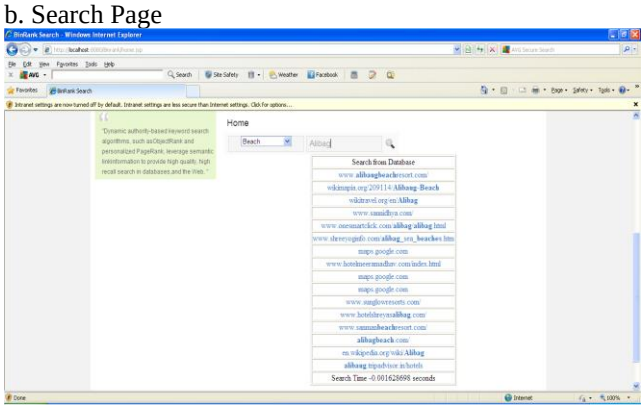
Sometimes user may wish to add his/her own category. This table is used to store the search category.
Here we give some of the screen shots of BinRank System.



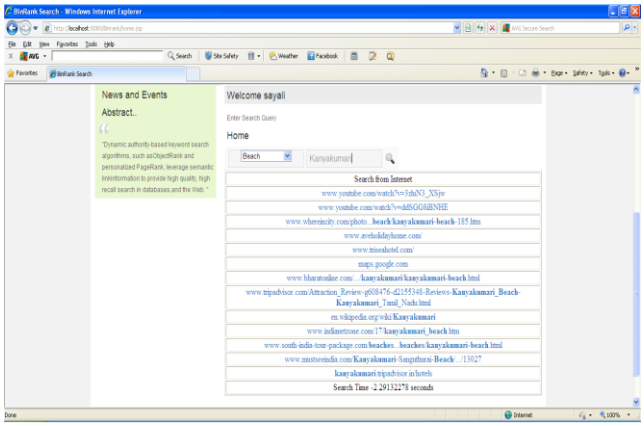
a. Index Page



b. Search Page

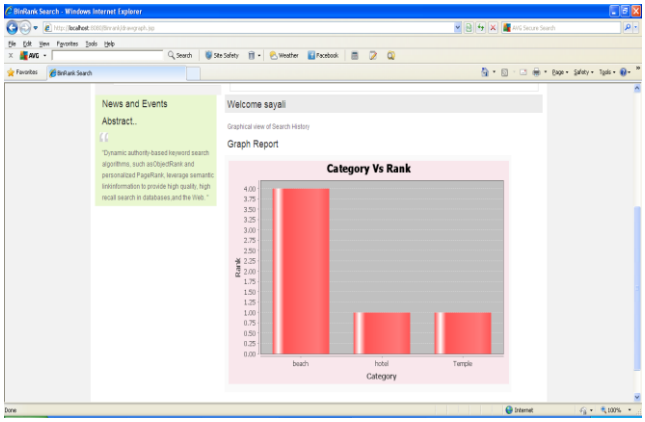


c. Results from database

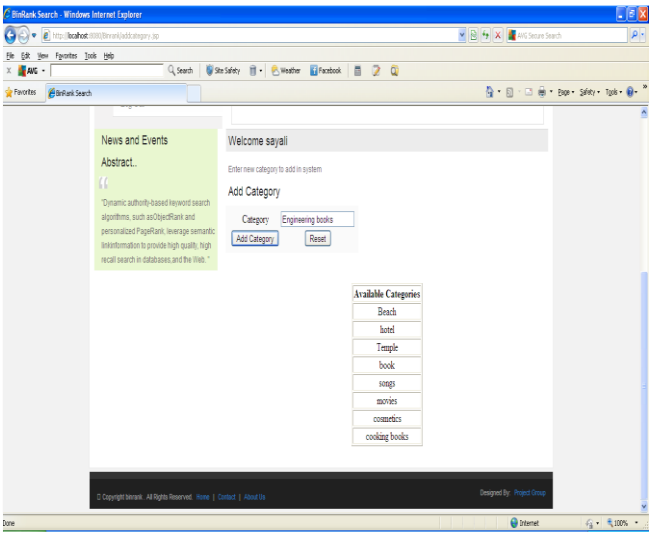


d. Results from Internet

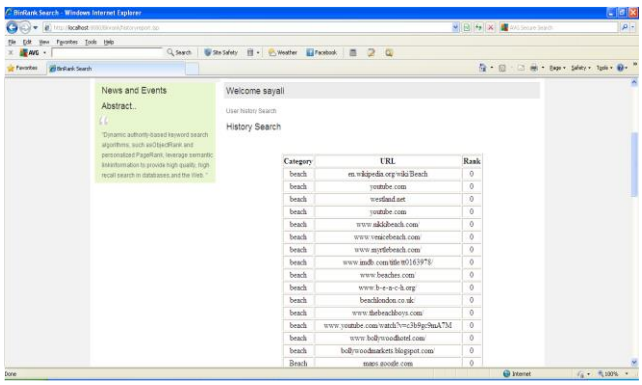
Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)



e. Graph of Category vs Rank



f. Adding a category



g. Searching History

Fig.2. Experimental Results

VII. ADVANTAGES OF BINRANK OVER OTHERS

- Highly secure compared to others. Provides authentication to each user.
- Privacy is provided to each user. Each user's detail is stored in account. So that user doesn't have to remember what links he has visited in past. All the history is maintained in account.
- All history information is stored in ranking order.
- Popular pages like google are very much overloaded because number of user access it at the same time. So if network is not fast or there is some problem in connection, it takes time to load single page. If the results are already stored in local cache, results can be retrieved fast from local cache instead of going through the network. After getting result from local memory, the url can be directly located into the address bar of the browser and information can be fetched directly.

VII. FUTURE EXTENSION

- This system uses Google's database as primary database, but in future we can merge database of other Engines also.
- This system, considers single level of search hierarchy. This system can be extended for unlimited depth of web page surfing.
- This system can be enhanced to provide all the available facilities for particular group (researching something) in more secure way.
- Other algorithms can be blended for optimizing the search using minimum path.
- Web crawler can be designed that will work supportively with existing system for providing better search.

IX. CONCLUSION

In this paper, we proposed BinRank as a practical solution for scalable dynamic authority-based ranking. It is based on partitioning and approximation using a number of materialized subgraphs. We showed that our tunable system offers a nice trade-off between query time and preprocessing cost. The BinRank algorithm groups co-occuring terms into number of bins for which materialized subgraphs are computed. The materialized subgraphs are computed offline by using ObjectRank itself. The intuition behind this approach is that a subgraph that contains all objects and links relevant to set of related terms should have all information needed to rank objects with respect to one of these terms. Our extensive experimental evaluation confirms this intuition. In future, we will study the impact of other keyword relevance measures, besides term co-occurrence such as thesaurus or ontologies, on the performance of BinRank . We will further explore better solution for queries whose random surfer starting points are provided by boolean conditions.

REFERENCES

1. T.H. Haveliwala, "Topic-Sensitive PageRank," Proc. Int'l World Wide Web Conf. (WWW), 2002.
2. G. Jeh and J. Widom, "Scaling Personalized Web Search," Proc. Int'l World Wide Web Conf. (WWW), 2003.
3. D. Fogaras, B. Ra´cz, K. Csaloga´ny, and T. Sarlo´ s, "Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments," Internet Math., vol. 2, no. 3, pp. 333-358, 2005.
4. A. Balmin, V. Hristidis, and Y. Papakonstantinou, "ObjectRank: Authority-Based Keyword Search in Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2004.
5. Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma, "Object-Level Ranking: Bringing Order to Web Objects," Proc. Int'l World Wide Web Conf. (WWW), pp. 567-574, 2005.
6. Ji-Lin, Ren Yong-jian, Zhang Wei, Xu Xiang-Hua, Wan Jian "Webs Ranking Model Based On PageRank Algorithm" IEEE transactions 2011.
7. Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp BinRank: Scaling Dynamic Authority-Based Search Using Materialized Subgraphs IEEE Transaction on Knowledge and Data Engineering VOL. 22, NO. 8, August 2010

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

8. Mandar Kale Mrs. P.Santhi Thilagam" DYNA-RANK: Efficient calculation and updation of PageRank" International Conference on Computer Science and Information Technology 2007
9. Yong Zhen Guo, Kotagiri Ramamohanarao and Laurence A. F. Park" Personalized PageRank for Web Page Prediction Based on Access Time-Length and Frequency" IEEE International conference on web intelligence 2007.

AUTHOR PROFILE

Prof. Tanaji D. Khadtare. received B.E. degree in 1988 in Electrical Engineering and M.E. degree in System Science and Automation from I.I.S.C. Banglore in 1996. He has been working in various engineering colleges for 24 years. He is currently working as a Professor and Head of Department of Information Technology in Sinhgad Institute of Technology and Science, Pune. His Research interest includes Data Mining, Algorithms and Artificial Neural Networks.

Prof. Prajakta R. Thakare received B.E.in Computer Technology from Nagpur University in 2005 and M.E. degree in Computer Science from University of Pune in 2009.She has been working in various engineering colleges for 8 years She is currently working as an Asst. Professor in Department of Information Technology in Sinhgad Institute of Technology and Science, Pune. Her Research interests are Information Retrieval, Computer Vision, Image processing and Multimedia.

Prof. S. A. J. Patel received B.E. in Information Technology in 2008 and M.E. degree in Computer Science & Engineering from Solapur University in 2012. She is currently working as an Asst. Professor in Department of Information Technology in Sinhgad Institute of Technology and Science, Pune. Her Research interests are Information Retrieval, Operating Systems, and Image processing .

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)