International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-6, August 2014

Effective Binrank for Scaling Dynamic Authority based Search with Materialized Sub Graphs

L Prasanna Kumar

Abstract. Dynamic authority-based keyword search algorithms, such as ObjectRank and personalized PageRank, leverage semantic link information to provide high quality, high recall search in databases, and the Web. Conceptually, these algorithms require a query time PageRank-style iterative computation over the full graph. In this paper we introduce BinRank system which approximates ObjectRank results by utilizing a hybrid approach inspired by materialized views in traditional query processing.

Keywords: World Wide Web, ObjectRank, subgraphs, BinRank

I. INTRODUCTION

The PageRank algorithm utilizes the Web graph link structure to assign global importance to Web pages. It works by modeling the behavior of a random Web surfer who starts at a random Web page and follows outgoing links with uniform probability. The PageRank 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. In particular, two algorithms have got a lot of attention: Personalized PageRank (PPR) and ObjectRank. PPR is a modification of PageRank that 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 fix point iterative computation over the entire Web graph, while it generates personalized search results [1].

ObjectRank 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. The resulting system is well suited for "high recall" search, which exploits different semantic connection paths between objects in highly heterogeneous data sets. For example, on the Wikipedia data set, the full dictionary precomputation would take about a CPU-year [2-5]. In this paper, we introduce a BinRank system that employs a hybrid approach where query time can be traded off for preprocessing time and storage.

II. LITERATURE SURVEY

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. To avoid the expensive iterative calculation at runtime, one can naively precomputes and materialize all the possible personalized PageRank vectors(PPV). 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.

Manuscript published on 30 August 2014.
***** Correspondence Author (s)
L. Prasanna Kumar*, Associate Professor, Department of CSE, Dadi Institute of Engineering & Technology, Visakhapatnam. (Andhra Pradesh), India. E-mail id: prasannakumar@dietakp.com

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

In this section, we give overview of HubRank that integrates the two approaches to improve the scalability of ObjectRank [6].

Hub-based approaches: Materialize only a selected subset of PPVs. Topic-sensitive PageRank suggests materialization of 16 PPVs of selected topics and linearly combining them at query time. The personalized PageRank computation enables a finer-grained personalization by efficiently materializing significantly more PPVs 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 [7-9].

Page Rank: The PageRank algorithm utilizes the Web graph link structure to assign global importance to Web pages. It works by modeling the behavior of a "random Web surfer" who starts at a random Web page and follows outgoing links with uniform probability. The PageRank score is independent of a keyword query. Recently, dynamic versions of the PageRank algorithm have become popular [10].

Personalized Page Rank: In particular, two algorithms have got a lot of attention: Personalized PageRank (PPR) for Web graph data sets and ObjectRank for graph-modeled databases. PPR is a modification of PageRank that 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 [11].

Object Rank: ObjectRank has successfully been applied to databases that have social networking components, such as bibliographic data and collaborative product design. 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 [12,13].

III. OBJECTIVE AND SYSTEM ARCHITECTURE

The main objective of this system is that employs a hybrid approach where query time can be traded off for preprocessing time and storage. BinRank closely approximates ObjectRank scores by running the same ObjectRank algorithm on a small subgraph, instead of the full data graph. In this paper, we are proposing the BinRank algorithm for the trade time of search. Our algorithm solves the time consuming problem in query execution. Time will be reduced because of cache storage and redundant query handling method.

3.1 System Architecture

During query processing stage (right side of figure 1), we execute the ObjectRank algorithm on the subgraphs instead of the full graph and produce high-quality approximations of p-k lists at a small fraction of the cost. In order to save preprocessing cost and storage, each MSG is designed to answer multiple term queries.

© Copyright: All rights reserved.

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. 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 request terms as shown in Fig 1. The ObjectRank module takes as input a set of bin posting lists B and the entire graph; EÞ 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 ObjectRankscore of MSG nodes.

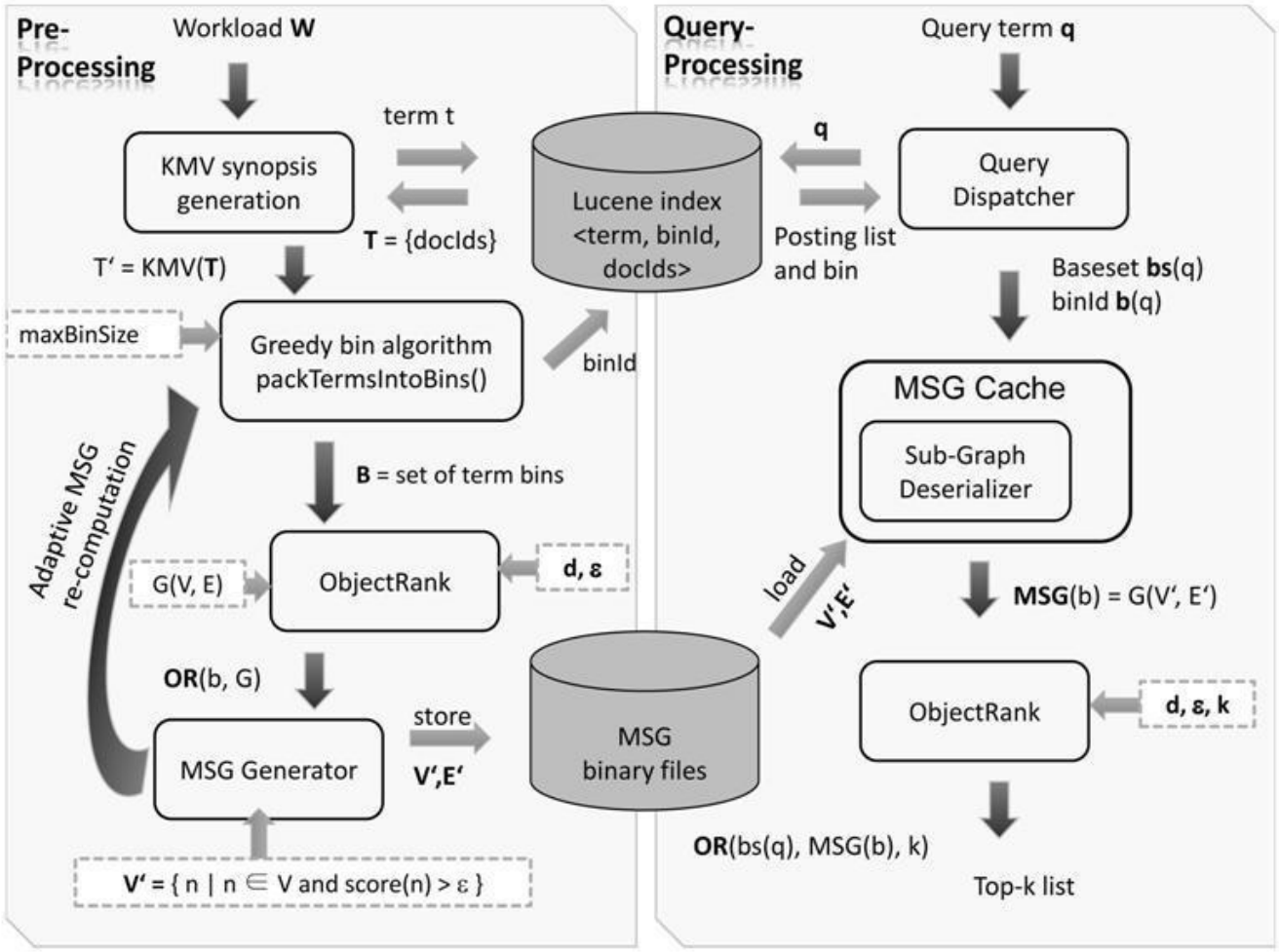

Fig 1 System architecture

3.2 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 Object Rank 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 Bin Rank 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 MS Gnodes V, and generating the corresponding set of edges E_0 only at query time.

3.3 Algorithm

Bin Computation Algorithm

Input: A set of workload terms W, with their posting lists
Output: A set of bins B

1. while W is not empty do
2. create a new empty bin b and empty cache of candidate terms C
3. pick term tϵW with the largest posting list size |t|
4. while t is not null do
5. add t to b, and remove it from W
6. compare a set of terms T that co-occur with t
7. for each t′ϵT do
8. insert (or update) mapping < t′, null> into C

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

```
9.  end
10. for each bestI:=0
11. for each mapping < c , i > ϵ C do
12. if i=null then i:=|b|
13. update mapping < c, I > in C
14. end if
15. union :=|b| + |c| - i
16. if union > maxBinSize then
17. remove < c,I > from C
18. else if i > bestI then bestI := i, t :=c
19. end if
20. end for each
21. if bestI = 0 then pick tϵW with maximum |t| ≤ maxBinSize - |b|
22. if no such t exists, t :=null
23. end if
24. end while
25. add completed b to B
26. end while
```

IV. IMPLEMENTATION

4.1 List of Modules

getting the membership login the users should made registration with our application. In registration we will get all the details about the users and it will be stored in a database to create membership.

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

Search Query Submission: Users query will be submitted in this module. Users can search any kind of things in our application when we connect with Internet. Users query will be processed based on their submission, and then it will produce the appropriate result.

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. By the way we are creating the index for speed up the search efficient and fast with the help of implementing BinRank algorithm.

User Registration: We are providing the facility to register new users. If anyone wants use our application, they should become a member of our application. To

BinRank Algorithm Implementation: We generate an MSG for every bin based on the intuition that a subgraph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms. Based on the index creation we need to generate the results for the users query.

Graph based on Rank: Graph will be generated based on the users queries submitted. This graph will represent the user search key word, number of websites produced for their search, how many times that websites occurred in the search result and the Rank for websites based on the user clicks. User may search the same key word again and again, so result may also produce as same URLs. At that user will click some of the URLs; based on their clicks the Rank will be calculated. Based on the Number of times URL occurrence, Rank and Keyword the Graph will generate as shown in Fig 2.

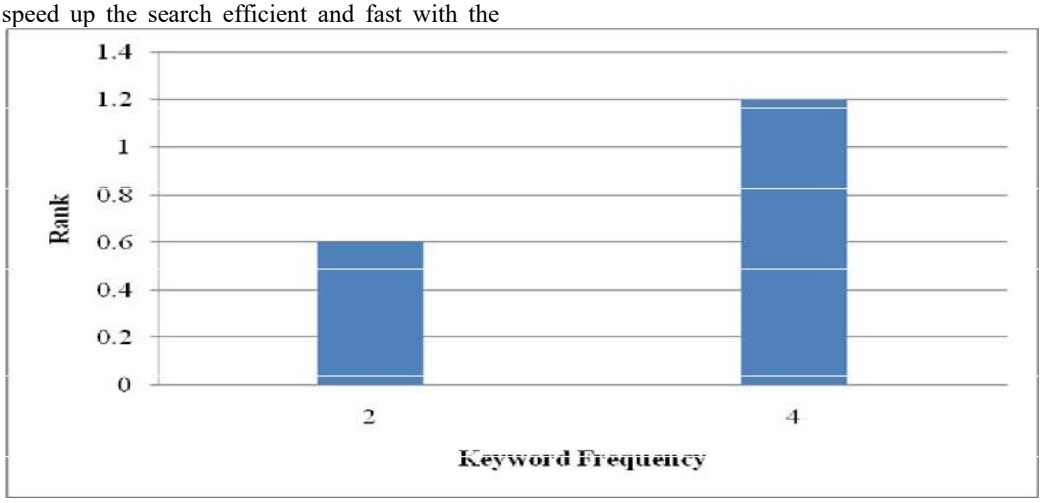

Fig 2 Keyword frequency vs rank

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

V. CONCLUSIONS

We present a performance comparison of BinRank over Monte Carlo style methods and HubRank. We implemented the Monte Carlo algorithm 4, “MC complete path stopping at dangling nodes,” introduced in [5] and HubRank [8] that combines a hub-based approach and a Monte Carlo method called fingerprint. For a given keyword query, the Monte Carlo algorithm simulates random walks starting from nodes containing the keyword. Within a specified number of walks, it samples exactly the same number of random walks per each starting point.. We used our workload keyword queries and executed the Monte Carlo algorithm with different total numbers of sampled walks. As the number of sampled walks increases, the algorithm generates higher quality top-k lists, which usually takes more time.

REFERENCES

1. s.brin, l.page,"the anatomy of a large-scale hypertextual web search engine",computer networks,vol.30, nos.1-7, pp. 107-117, 1998.
2. t.h.haveliwala,"topic-sensitivepagerank,"proc.int’l world wide web conf.(www),2002.
3. g.jeh, j.widom,”scaling personalized web search,”proc.int’l world wide web conf.(www),2003.
4. d.fogaras, b.racz,k.csalogany,and .sarlos,"towards scaling fully personalized pagerank: algorithms, lower bounds,and experiment", internet math.,vol.2,no.3,pp.333-358,2005.
5. k.avrachenkov,n.litvak,d.nemirovsky, n.osipova,"monte carlo methods in pagerank computation:when one iteration is sufficient", siam j.numerical analysis,vol.45,no.2, pp.890-904,2007.
6. a.balmin,v.hristidis, y.papakonstantinou,"objectrank:authority-based keyboard search in databases", proc.int’l conf.very large data bases (vldb),2004.
7. znie , y. zhang , j .r . wen , w. y. ma , " object - level ranking:bringing order to web objects", proc.int’l world wide web conf.(www),pp.567-574,2005.
8. s.chakrabarti,"dynamic personalized pagerank in entityrelations graphs", proc.int’l world wide web conf.(www),2007.
9. h.hwang,a.balmin,h.pirahesh, b.reinwald,”information discovery in loosely integrated data,”proc.acm sigmod, 2007.
10. v.hristidis,h.hwang, y.papakonstantinou,"authority-based keyword search in databases,”acm trans. database systems,vol.33, no.1, pp. 1-40,2008.
11. m.r.garey, d.s. johnson,"a 71/60 theoremfor bin packing,"j.complexity,vol.1,pp.65-106, 1985.
12. k.s.beyer,p.j.haas,b.reinwald,y.sismanis, r.gemulla,"on synopses for distinct-value estimation under multiset operations,"proc.acm sigmod, pp .199-210, 2007.
13. j.t.bradley, d.v.de jager,w.j.knottenbelt, a.trifunovic, "hypergraph partitioning for faster parallel pagerank computation , ” EPEW,pp. 155-171, 2005.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.