Middle-East Journal of Scientific Research 20 (3): 347-352, 2014
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2014.20.03.11315

Search of Data Patterns by Ranking Object and Constructing Bins Using Materialized Sub Graph

P. Ramya

Department of CSE, Bharath University, India

Abstract: To perform a key word search from the database the object rank algorithms and page rank algorithms are used. These are general search algorithms basically used in all search engines, these algorithms uses iterative computation over a full graph hence this computation becomes expensive for large graphs and also had expensive preprocessing. To make the computation simpler here the bin ranking and hub ranking are introduced. Here bin rank involves in generating the sub graphs by partitioning all the term based on their co-occurrence. The intuition is that a sub graph that contain 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. The bin rank can achieve sub second query execution time on the English Wikipedia data set. The general object rank search will approximate the search result on the graph. Thus this experimental evaluation in depth explains the trade-off between query execution, time and quality of results. The hub rank help to minimize the search engine traffic by ranking the recently viewed path.

Key words: Bin rank · Hub rank

INTRODUCTION

Search of any data requires three basic algorithms that are used to minimize the search time and increase the accuracy of the search that includes Page ranking algorithm, Object ranking algorithm, personalized page ranking algorithm [1-5].

Page Ranking Algorithm: The Page Rank 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 [6-10].

Object Ranking Algorithm: Object Rank 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 [11-15].

Personalized Page Ranking: PPR is a modification of Page Rank 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.

The Page Rank 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 Page Rank score is independent of a keyword query. Recently, dynamic versions of the Page Rank 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 Page Rank (PPR) for Web graph data sets and Object Rank for graph-modeled databases.PPR is a modification of Page Rank 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

Corresponding Author: P. Ramya, Department of CSE, Bharath University, India.

Ranking of Data by Search Engine

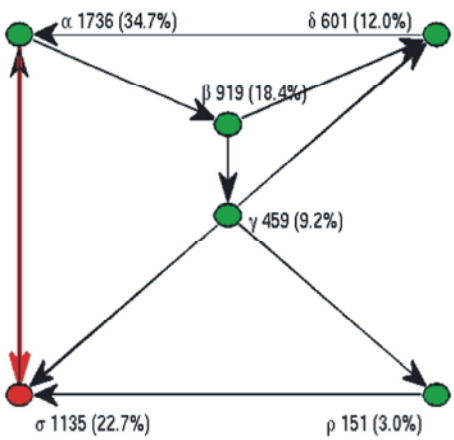

Fig. 1: Ranging of Data By Search Engine.

the entire Web graph, while it generates personalized search results. Therefore, the issue of scalability of PPR has attracted a lot of attention. Object Rank extends (personalized) Page Rank to perform keyword search in databases. Object Rank 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. Object Rank has successfully been applied to databases that have social networking components, such as bibliographic data and collaborative product design.

However, Object Rank suffers from the same scalability issues as personalized Page Rank, as it requires multiple iterations over all nodes and links of the entire database graph. The original Object Rank system has two modes: online and offline. The online mode runs the ranking algorithm once the query is received, which takes too long on large graphs. For example, on a graph of articles of English Wikipedia1 with 3.2 million nodes and 109 million links, even a fully optimized in-memory implementation of Object Rank takes 20-50 seconds to run the offline mode, Object Rank recomputed top-k results for a query workload in advance. This precipitation is very expensive and requires a lot of storage space for recomputed results. Moreover, this approach is not feasible for all terms outside the query workload that a user may search for, i.e., for all terms in the data set dictionary. For example, on the same Wikipedia data set, the full dictionary precomputation would take about a CPU-year [16].

Related Work: The issue of scalability of PPR has attracted a lot of attention. PPR performs a very expensive fix point iterative computation over the entire graph, while it generates personalized search results.

To avoid the expensive iterative calculation at runtime, one can naively precompute and materialize all the possible personalized Page Rank 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 Hub Rank that integrates the two approaches to improve the scalability of Object Rank. 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. Topic-sensitive Page Rank suggests materialization of 16 PPVs of selected topics and linearly combining them at query time. The personalized Page Rank 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 Page Rank, 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 Page Rank on any arbitrary preference set with maintaining just a small amount of recomputed results, Fingerprint algorithm that simulates the random walk model of Page Rank and stored the ending nodes of sampled walks. 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 [17].

General Issues in Ranking Algorithm: In general the ranking is always made in random by the various ranking algorithm that are mentioned above, the major drawback of these ranking algorithms are reduced scalability, more time consumption. Even though the Object Rank has successfully been applied to databases that have social networking components, such as bibliographic data and collaborative product design it does not possess the

property of scalability in proper These problems can be overcome by Employing a hybrid approach where query time can be traded off for preprocessing time and storage. The search graph, where the search take place in random must be made in particular. The graph must be segmented into various sub graphs and search of keyword must be allowed. We introduce a Bin Rank system that employs a hybrid approach where query time can be traded off for preprocessing time and storage. Bin Rank closely approximates Object Rank scores by running the same Object Rank algorithm on a small sub graph, instead of the full data graph. The sub graphs are recomputed offline. The recompilation can be parallelized with linear scalability. There are two dimensions to the sub graph precomputation problem: how many sub graphs to precompute? How to construct each sub graph that is used for approximation? The intuition behind our approach is that a sub graph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects w.r.t. one of these terms. Query performance is highly correlated to the size of the sub graph, which, in turn, is highly correlated with the number of documents in the bin. Thus, normally, it is sufficient to create bins with a certain size limit to achieve a specific target running time

Maintaining Object Rank: Object Rank performs top-k relevance search over a database modeled as a labeled directed graph. The data graph G (V, E) models objects in a database as nodes and the semantic relationships between them as edges. For a given query, Object Rank returns top-k objects relevant to the query. We first describe the intuition behind Object Rank, introduce the Object Rank equation and then, elaborate on important calibration factors. Object Rank returns top-k search results for a given query using both the content and the link structure in G. Since it utilizes the link structure that captures the semantic relationships between objects

Relevant Subgraphs: Our goal is to improve the scalability of Object Rank while maintaining the high quality of top-k result lists. We focus on the fact that Object Rank does not need to calculate the exact full Object Rank vector r to answer a top-k keyword query (K _ java j). We identify three important properties of Object Rank vectors that are directly relevant to the result quality and the performance of Object Rank.

First, for many of the keywords in the corpus, the number of objects with non-negligible Object Rank values is much less than jV j. This means that just a small portion of G is relevant to a specific keyword. Here, we say that an Object Rank value of v, rðvÞ is non-negligible if rðvÞ is above the convergence threshold. The intuition for applying the threshold is that differences between the scores that are within the threshold of each other are noise after Object Rank execution. Thus, scores below threshold are effectively indistinguishable from zero and objects that have such scores are not at all [18].

BIN Construction: As outlined above, we construct a set of MSGs for terms of a dictionary or a workload by partitioning the terms into a set of term bins based on their co-occurrence. We generate an MSG for every bin based on the intuition that a sub graph 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. There are two main goals in constructing term bins. First, controlling the size of each bin to ensure that the resulting sub graph is small enough for Object Rank to execute in a reasonable amount of time. Second, minimizing the number of bins to save the preprocessing time. After all, we know that precomputing Object Rank for all terms in our corpus is not feasible. To achieve the first goal, we introduce a maxBinSize parameter that limits the size of the union of the posting lists of the terms in the bin, called bin size. As discussed above, Object Rank uses the convergence threshold that is inversely proportional to the size of the base set, i.e., the bin size in case of sub graph construction.

Thus, there is a strong correlation between the bin size and the size of the materialized sub graph. As show in Section 8, the value of maxBinSize should be determined by quality and performance requirements of the system [19].

Query Processing: A given keyword query q, the query dispatcher retrieves from the Lucerne 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 mapped 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. The Object Rank module gets the in-memory instance of MSG, the base set and a set of Object Rank calibrating parameters:

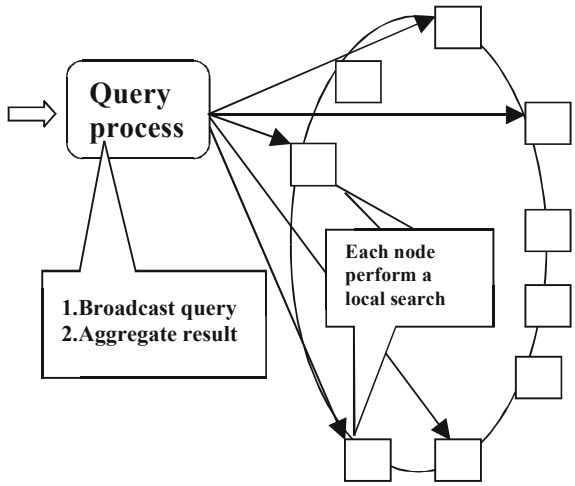

Fig. 2: The Bin Rank Query Processing.

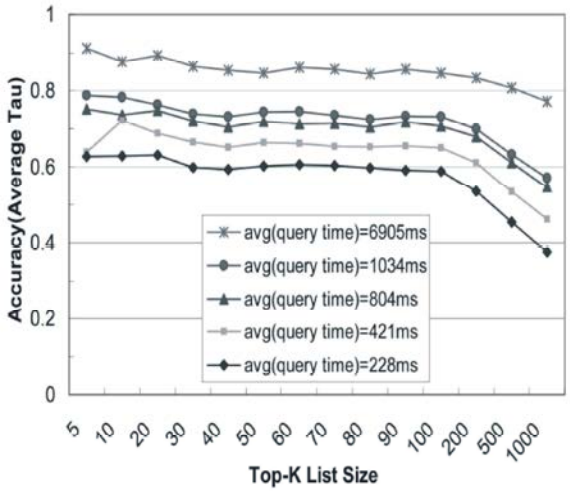

Fig. 3: Performance Comparison of Bin Rank

The damping factor d, the convergence threshold, the number of top-k list entries k. Once the Object Rank scores are computed and sorted, the resulting document ids are used to retrieve and present the top-k objects to the user.

HUB Construction: Hub ranking is mainly involved in maintaining the traffic in the web during the search engine performs the search. The hub construction involves few steps. The first thing to do when making a high traffic hub is picking the keywords. Search may cause competition. If there is little competition for a title is likely to bring in traffic.

Then Choose the right category, the category depends on the groups of item. There are basically two categories fits your topic, too crowded. Interlinking the hubs. The easiest way to get started on that is to interlink your own hubs.These are the three steps involved in preprocessing, the successful construction of these will provide a scalable search engine.

Performance Comparison of Bin Rank with Monte Carlo Method and Hub Rank In this section, we present a performance comparison of Bin Rank over Monte Carlo style methods and Hub Rank. We implemented the Monte Carlo algorithm, "MC complete path stopping at dangling nodes," introduced in and Hub Rank 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. The authority score of a node is the total number of visits to the node divided by the total number of visits. During the preprocessing stage (left side of figure), we generateMSGs. During query processing stage (right side of figure), we execute the Object Rank algorithm on the sub graphs instead of the full graph and produce high-quality approximations of top-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. We observed in the Wikipedia data set that a single

MSG can be used for 330-2,000 terms, on average MSG generation. Once the bins are constructed, we generate an MSG for each bin. For our Wikipedia data set, we generated a comprehensive set of MSGs with 24 combinations of the two parameters maxBinSize and _. For each combination, we measure the performance of Bin Rank, i.e., the query time and the quality of top-k lists. maxBinSize determines the number of bins to be constructed and thus, the number of MSGs generated. The construction time and average size go up with the maxBinSize. Intuitively, the larger the base set, the more objects will be related to it. And the more objects have nontrivial scores, the more iterations it will take the Object Rank algorithm to reach the fixpoint. Quality measures. For a given keyword query, Bin Rank generates an approximate top-k list using the corresponding MSG. The exact top-k list is obtained by executing Object Rank on Gwiki with small _ ¼ 1:0E-4. The two lists are compared using the same three quality measures as in: relative aggregated goodness (RAG), precision at K and Kendall's Let ORðkw;KÞ and BRðkw;KÞ denote the accurate top-k list by Object Rank and the approximate top-k list by Bin Rank for a given keyword kw. In our experiments, both

top-k lists are lists of Wikipedia article IDs sorted by the authority score. Let ORScoreðn; kwÞ denote the exact keyword-specific authority score of a node n computed by Object Rank.

CONCLUSION

In this paper, we proposed Bin Rank as a practical solution for scalable dynamic authority-based ranking. It is based on partitioning and approximation using a number of materialized sub graphs. We showed that our tunable system offers a nice trade-off between query time and preprocessing cost. We introduce a greedy algorithm that groups co-occurring terms into a number of bins for which we compute materialized sub graphs. Note that the number of bins is much less than the number of terms. The materialized sub graphs are computed offline by using Object Rank itself. The intuition behind the approach is that a sub graph 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. Our extensive experimental evaluation confirms this intuition. For future work, we want to study the impact of other keyword relevance measures, besides term co-occurrence, such as thesaurus or ontologism, on the performance of Bin Rank. By increasing the relevance of keywords in a bin, we expect the quality of materialized sub graphs, thus the top-k quality and the query time can be improved. We also want to study better solutions for queries whose random surfer starting points are provided by Boolean conditions. And ultimately, although our system is tunable, the configuration of our system ranging from number of bins, size of bins and tuning of the Object Rank algorithm itself (edge weights and thresholds) is quite challenging and a wizard to aid users is desirable. To further improve the performance of Bin Rank, we plan to integrate Bin Rank and Hub Rank by executing Hub Rank on MSGs Bin Rank generates. Currently, we use the Object Rank algorithm on MSGs in query time. Even though Hub Rank is not as scalable as Bin Rank, it performs better than Object Rank on smaller graphs such as MSGs. In this way, we can leverage the synergy between Bin Rank and Hub Rank [20-25].

REFERENCES

1. Brin, S. and L. Page, 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine, Computer Networks, 30: 1-7, 107-117.
2. Haveliwala, T.H., 2002. Topic-Sensitive Page Rank, Proc. Int'l World Wide Web Conf. (WWW).
3. Jeh, G. and J. Widom, 2003. Scaling Personalized Web Search, Proc.Int'l World Wide Web Conf. (WWW).
4. Fogaras, D., B. Ra´cz, K. Csaloga´ny and T. Sarlo´s, 2005. Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds and Experiments, Internet Math., 2(3): 333-358.
5. Avrachenkov, K., N. Litvak, D. Nemirovsky and N. Osipova, 2007. Monte Carlo Methods in PageRank Computation: When One Iteration Is Sufficient, SIAM J. Numerical Analysis, 45(2): 890-904.
6. Balmin, A., V. Hristidis and Y. Papakonstantinou, 2004. Object Rank: Authority-Based Keyword Search in Databases, Proc. Int'l Conf. Very Large Data Bases (VLDB).
7. Nie, Z., Y. Zhang, J.R. Wen and W.Y. Ma, 2005. Object-Level Ranking: Bringing Order to Web Objects, Proc. Int'l World Wide Web Conf. (WWW), pp: 567-574.
8. Chakrabarti, S., 2007. Dynamic Personalized Page Rank in Entity- Relation Graphs, Proc. Int'l World Wide Web Conf. (WWW).
9. Hwang, H., A. Balmin, H. Pirahesh and B. Reinwald, 2007. Information Discovery in Loosely Integrated Data, Proc. ACM SIGMOD.
10. Hristidis, V., H. Hwang and Y. Papakonstantinou, 2008. Authority-Based Keyword Search in Databases, ACM Trans. Database Systems, 33(1): 1-40.
11. Kendall, M., 1955. Rank Correlation Methods. Hafner Publishing Co.
12. Garey, M.R. and D.S. Johnson, 1985. A 71/60 Theorem for Bin Packing, J. Complexity, 1: 65-106.
13. Beyer, K.S., P.J. Haas, B. Reinwald, Y. Sismanis and R. Gemulla, 2007. On Synopses for Distinct-Value Estimation under Multiset Operations, Proc. ACM SIGMOD, pp: 199-210.
14. Bradley, J.T., D.V. de Jager, W.J. Knottenbelt and A. Trifunovic, 2005. Hypergraph Partitioning for Faster Parallel Page Rank Computation, Proc. Second European Performance Evaluation Workshop (EPEW), pp: 155-171.
15. Cho, J. and U. Schonfeld, 2007. Rankmass Crawler: A Crawler with High Page Rank Coverage Guarantee, Proc. Int'l Conf. Very Large Data Bases (VLDB).
16. Kerana Hanirex, D. and K.P. Kaliyamurthie, 2013. Multi-classification approach for detecting thyroid attacks, International Journal of Pharma and Bio Sciences, 4(3): B1246-B1251.

17. Khanaa, V., K. Mohanta and T. Saravanan, 2013. Comparative study of uwb communications over fiber using direct and external modulations, Indian Journal of Science and Technology, 6(6): 4845-4847.
18. Kumar, Giri, R. and M. Saikia, 2013. Multipath routing for admission control and load balancing in wireless mesh networks, International Review on Computers and Software, 8(3): 779-785.
19. Kumarave, A. and K. Rangarajan, 2013. Routing alogrithm over semi-regular tessellations, 2013 IEEE Conference on Information and Communication Technologies, ICT 2013.
20. Kumarave, A. and K. Rangarajan, 2013. Algorithm for automaton specification for exploring dynamic labyrinths", Indian Journal of Science and Technology, 6(6).
21. Shafaq Sherazi and Habib Ahmad, 2014. Volatility of Stock Market and Capital Flow Middle-East Journal of Scientific Research, 19(5): 688-692.
22. Kishwar Sultana, Najm ul Hassan Khan and Khadija Shahid, 2013. Efficient Solvent Free Synthesis and X Ray Crystal Structure of Some Cyclic Moieties Containing N-Aryl Imide and Amide, Middle-East Journal of Scientific Research, 18(4): 438-443.
23. Pattanayak, Monalisa and P.L. Nayak, 2013. Green Synthesis of Gold Nanoparticles Using Elettaria cardamomum (ELAICHI) Aqueous Extract World Journal of Nano Science and Technology, 2(1): 01-05.
24. Chahataray, Rajashree and P.L. Nayak, 2013. Synthesis and Characterization of Conducting Polymers Multi Walled Carbon Nanotube-Chitosan Composites Coupled with Poly (P-Aminophenol) World Journal of Nano Science and Technology, 2(1): 18-25.
25. Parida, Umesh Kumar, S.K. Biswal, P.L. Nayak and B.K. Bindhani, 2013. Gold Nano Particles for Biomedical Applications World Journal of Nano Science and Technology, 2(1): 47-57.