Efficient Web Object Caching through Probabilistic Modeling

T S Bhagavath Singh, S Chitra

Abstract: ***Web object search engines are a new breed of Web IR systems, wherein, the focus is to provide relevant results, such that, the result set has limited influx of unnecessary information. To achieve this said goal, the document information is extracted, and reorganized in the Web object repository, wherein, Web objects are logical entities such as-cities, authors, stadiums etc. Caching has become an integral part of modern Web search engines. Many efficient caching schemes have been proposed for this domain. However, the same attention is lacking for Web object search engines. The contemporary solution in the literature is quite primitive, and lacks the required effectiveness. In this work, a new caching mechanism based on probabilistic model is presented. The proposed caching mechanism is evaluated empirically along with the contemporary technique. Empirical results exhibit superior performance of the proposed scheme in-terms of cache hit ratio and query execution time.***

Keywords:Web Object **Search, Caching, Query Clustering.**

I. INTRODUCTION

A. Overview

Modern Web search engines focus on retrieving the relevant Web documents for the user query. However, the Web documents might contain numerous Web object information, and the user might require information for only a single or few Web objects. In such situation, the user has to browse manually to search for the specific object information in the retrieved Web documents, which can decrease the overall effectiveness of the Web search engine. To circumvent this issue, a new breed of IR systems called Web object search engines has been proposed. Here, the required objects are first identified, the relevant information for the corresponding objects are extracted from different Web documents, and stored in object repository. The object information is structured in-terms of tuples and attributes.

The two earliest and popular Web object search engines are-Windows Live Product Search and Libra Academic Search, which provided object search facility for different products and academic publications respectively.

B. Overview on IR caching

Generally, Caching refers to storage of certain elements in the main memory which have high potential to be retrieved large number of times by the future queries. This strategy aids in quicker response time for query execution. If the desired result for a query is not found in the cache, the corresponding data repository is accessed, and this situation is denoted as Cache Miss. The ideal caching strategy has to minimize cache miss ratio.

Caching mechanisms have become an important aspect of many contemporary IR systems [1-4], in-order to improve their query response time. Most of these caching mechanisms store queries which are submitted with high frequency, and their corresponding result set.

C. Motivation

The size of Web is exploding, and this Big Data issue has enforced new challenges for Web search engines to provide effective results in limited response time. Since, Web object search engines are also directly influenced by burgeoning Web, it is essential to achieve the required efficiency in object query execution. Caching has been popular in achieving the efficiency goals in many IR systems, and also provides an attractive option to improve the object query execution efficiency in Web object search engines. The contemporary solution available [5] is extremely primitive, and does not achieve required effectiveness, which was revealed in the empirical evaluations performed in this work. Hence, it is extremely essential to design an effective caching mechanism for Web object search engines to cater the future Big Data issue.

D. Paper Contributions

In this work, the following contributions are made:

A new distributed Web object caching mechanism is proposed. The queries which have high potential to get resubmitted in future are selected through the utilization of a probabilistic model. The selected queries are grouped based on their result set similarity, wherein, the queries in each group may have good number of overlapping results. The query group creation is achieved through correlation coefficient function. Each group is assigned scores based on their potential to be submitted in future. The highest ranked groups are selected to be placed in the cache memory.

The proposed cache mechanism is implemented inside custom built Web object search engine. The object repository is created by using real world data sets. The empirical results demonstrate orders of magnitude improvement in query response time, and reduction in cache miss ratio compared to contemporary technique [5].

Revised Manuscript Received on May 28, 2019.

First Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.
Second Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.
Third Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.

This paper is organized as follows: Section 2 describes the related work in this area. Section 3 presents the proposed Web object caching mechanism. Section 4 outlines the empirical results. Finally, Section 5 concludes with future work in this area.

II. Related Work

Data record extraction is performed by numerous applications on Web pages and Web databases [6]. Some of the data extraction techniques identify the required data records or blocks in different Web documents, and perform extraction to store the extracted objects in the corresponding repository. In many cases, the extracted elements do not have explicit labels, and ambiguity might arise in deciding the proper label. Many label resolving techniques have been proposed, and recently, Conditional Random Field has become a popular technique to achieve this goal.

Data extraction techniques are also employed on XML documents. Since, XML documents have tree structure; many queries only consider certain sub-trees. To cater such queries, a new XML document repository is created by extracting suitable XML document sub-trees. Some of the techniques proposed in [7, 8] efficiently achieve the said goal.

Some of the distributed IR systems [9, 10] have similarities with Web object search engine, due to their utilization of document extraction and integration process. However, these extraction and integration techniques cannot be directly applied to Web object search engines due to their specific nature.

In [11] ranking functions to score different Web objects in the result set of a query was proposed. This ranking function combined two models attribute level model and record level model which considers the Web object information in-terms of attributes and records respectively. It was argued in [11] that, combining both attribute and data record model for scoring Web objects provides better effectiveness.

In [12] location specific search technique for Web objects was proposed, wherein, the Web objects relevant to a specific location demanded by the query are retrieved. A specialized data structure denoted as IR Tree was proposed to achieve the spatial search of Web objects. In [13], geographical labeling technique was proposed for Web objects, wherein, the Web objects which do not have specified location labels are subjected to labeling process. The presented geographical labeling technique utilized Gaussian Mixture Model.

Semantic search is accomplished through a special class of IR systems known as Web of Data [14-16]. Here, the users can frame their queries in normal communicating sentences. The objects of retrieval are denoted as Entities, and they can be linked together based on their semantic relationships. This entity network has inspired the name Web of Data. Formal models were presented in [17] to assess the result quality for Web of Data. In [18], the similarity between two entities was measured through a metric called Co-reference.

Caching based IR system was proposed in [1]. If a cache miss occurs; the partial collection which contains part of the document repository, and which can substantially satisfy the user query requirements is accessed. Basically, two level caching is employed, and if both caches fail to deliver the required result set, the document repository is accessed.

Proxy servers were utilized to implement caching in IR systems [2]. The proxy server stores most of the popular documents, and it is located between the client and the actual server. The main goals of the proxy server are to reduce query execution latency by acting as a cache, and load reduction in the original servers.

Web access through mobile systems has become a commonplace. Mobile Caching is a technique in which the cache is created in the mobile system. It was argued in [4]; the traditional caching mechanisms are unsuitable due to efficiency issues in mobile environments, and a mobile caching framework was proposed, which flushes out the cache content for a specific location as soon as the user exits the location. Also, the proposed caching scheme considered the correlation between the locations for cache replacement.

Introductory work on Web object caching was presented in [5]. Here, the Web objects that are frequently accessed are stored in the cache. The main issue with this approach is that{for a new query only some of the result set Web objects might be located in the cache, which can limit the cache effectiveness. It is clear from the presented Related Work; until now, focus on caching in Web object search engines has got limited attention.

III. Web Object Caching Technique

A. Architecture

Figure 1 illustrates the architecture of the proposed caching technique. Here, Query processing component is involved in query pre-processing; to identify the different object data sources for retrieving the results. There are m Web object sources containing different object information. Each object source contains corresponding cache storage. It is clear that, the proposed architecture is distributed in nature, and can scale to Big Data environments. Also, each data source has corresponding cache storage.

Due to the distributed architecture, the result set of many queries might contain Web objects which belong to different object data sources, and due to the availability of good bandwidth in contemporary distributed IR systems; it is assumed that, the cache corresponding to a speci c object data source can store Web objects belonging to other data sources.

Let, D1, D2, …Dm indicate the number of data sources present in the considered Web object search engine. The cache for Di($1 \leq i \leq m$) is indicated as Ci. The storage capacity of Ci is indicated by size(Ci). The cache capacity of the entire Web object search engine is indicated by size(C), and represented in Equation 1. Here, C indicates the cache for the entire Web search engine.

$$size(C) = \sum_{i=1}^{m} size(C_i) \quad (1)$$

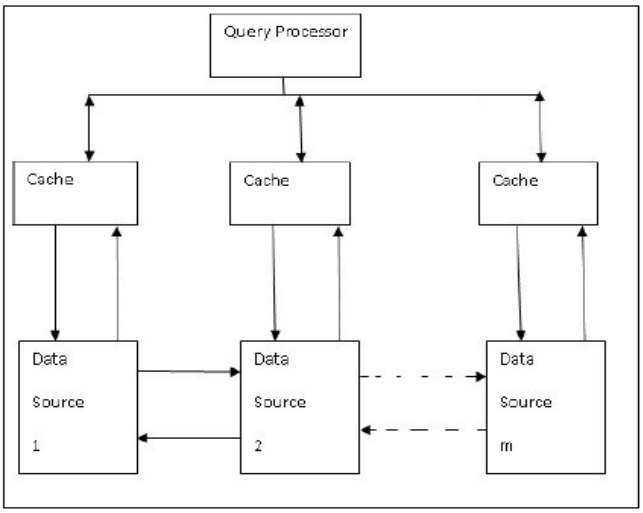

Fig. 1. Web Object Search Engine Architecture

B. *Probabilistic Model for Query Score Function*

Each distinct query appearing in the considered section of the query log is as-signed a score to indicate its reappearance potential in the future. The queries which have high reappearance potential and their corresponding result set are considered to be stored in the cache. The query log for time interval T , which is quite recent is utilized. Here, T is divided into nt equal time intervals indicated by (T1,T2, …Tnt) and each such divided time interval is termed as Measured Time Interval(MTI). Here, MTI is considered as a basic unit of measurement. Let, there be k distinct queries present in the query log, which are indicated as (Q1,Q2,…Qk). The frequency of Qj($1 \le j \le k$) appearing in the MTI Tp($1 \le p \le nt$); based on the empirical judgment is considered to be Poisson distributed with parameter j, and this case is represented in Equation 2. Here, f reqp(Qj) indicates the frequency of Qj appearing in the MTI Tp. Also, (f req1(Qj), f req2(Qj), ….) corresponding to each exclusive MTI are considered Identical and Independently Distributed (IID) Poisson random variables.

$$freq_p(Q_j) \sim Poisson(\lambda_j) \tag{2}$$

By using the property of IID variables in estimating their expectation, the parameter λ_j is estimated as represented in Equation 3.

$$\lambda_j \approx \frac{\sum_{i=1}^{n_t} freq_i(Q_j)}{n_t} \tag{3}$$

The score for Qj is represented in Equation 4. Here, score(Qj) indicates the score of Qj. Higher scores re ect the potential of Qj appearing in future with high frequency.

$$score(Q_j) = \lambda_j \tag{4}$$

Consider a query group Gb which contains jGbj number of queries indicated by (Qb1, Qb2,…QbjGbj). The score for this group is represented in Equation 5. Here, group score(Gb) indicates the group score of Gb, and higher group score indicates that this group contains queries which have high potential to be submitted in future.

$$group\_score(\mathbf{G_b}) = \frac{\sum_{j=1}^{|\mathbf{G_b}|} score(Q_{\mathbf{b}j} \in \mathbf{G_b})}{|\mathbf{G_b}|} \tag{5}$$

C. *Query Feature Vector*

The query feature vector aids in calculating correlation with other queries. Let, n Web objects be present in the object repository, which includes all the individual object data sources. These Web objects are identified as (O1,O2, …On). The query feature vector for Qj is represented in Equation 6. Here, Vj indicates the query feature vector for Qj and vj(Oi)($1 \le i \le n$) is represented in Equation 7.

$$\mathbf{V_j} = \begin{bmatrix} v_j(O_1) \\ v_j(O_2) \\ . \\ . \\ . \\ v_j(O_n) \end{bmatrix} \tag{6}$$

$$v_j(O_i) = \begin{cases} 1, & \text{if } O_i \in \text{ to the result set of } Q_j \\ 0, & otherwise \end{cases} \tag{7}$$

The group feature vector represents the query group. The group feature vector of Gb is represented in Equation 8. Here, jj represents the logical or operator, wherein the components in a specific position for two vectors are subjected to the logical or operation, group vector(Gb) indicates the group feature vector of Gb, and (Vb1, Vb2, …VbjGbj) indicate the query feature vectors of all the queries (Qb1, Qb2, …QbjGbj), which 2 Gb.

$$\text{group\_vector}(\mathbf{G_b}) = \mathbf{V_{b1}} || \mathbf{V_{b2}} || .... || \mathbf{V_{b|G_b|}} = \begin{bmatrix} v_{\mathbf{G_b}}(O_1) \\ v_{\mathbf{G_b}}(O_2) \\ . \\ . \\ . \\ v_{\mathbf{G_b}}(O_n) \end{bmatrix} \tag{8}$$

The size of Gb is represented in Equation 9. Here, size(Oi)($1 \le i \le n$) indicates the size of Web object Oi and size(Gb) indicates the size of Gb. The Equation 9 indicates that, if any Web object Oi is part of the result set of many queries that 2 Gb, then, only a single copy of Oi will be considered inside the group Gb.

$$size(\mathbf{G_b}) = \sum_{i=1}^{n} v_{\mathbf{G_b}}(O_i) \times size(O_i) \tag{9}$$

D. *Query Correlation Function*

The correlation function detects similarity or distance between queries. The correlation function for the queries Qi and Qj, where, i 6= j is represented in Equation 10. Here, corr(Vi, Vj) indicates the correlation function value, var(Vi) and var(Vj) indicate the variance of the values present in Vi and Vj respectively, and they are represented in Equations 11 and 12 respectively. Here, Vi and Vj indicate the

sample mean of Vi and Vj respectively, and which are represented in Equations 13 and 14 respectively. The function covar(Vi, Vj) indicates the covariance of the values present in Vi and Vj, and it is represented in Equation 15.

$$corr(\mathrm{V_i}, \mathrm{V_j}) = \frac{covar(\mathrm{V_i}, \mathrm{V_j})}{\sqrt{var(\mathrm{V_i})var(\mathrm{V_j})}} \quad (10)$$

$$var(\mathrm{V_i}) = \frac{1}{n-1}\sum_{r=1}^{n}(v_i(O_r) - \overline{\mathrm{V}}_\mathrm{i})^2 \quad (11)$$

$$var(\mathrm{V_j}) = \frac{1}{n-1}\sum_{r=1}^{n}(v_j(O_r) - \overline{\mathrm{V}}_\mathrm{j})^2 \quad (12)$$

$$\overline{\mathrm{V}}_\mathrm{i} = \frac{\sum_{r=1}^{n} v_i(O_r)}{n} \quad (13)$$

$$\overline{\mathrm{V}}_\mathrm{j} = \frac{\sum_{r=1}^{n} v_j(O_r)}{n} \quad (14)$$

$$covar(\mathrm{V_i}, \mathrm{V_j}) = \frac{1}{n-1}\sum_{r=1}^{n}(v_i(O_r) - \overline{\mathrm{V}}_\mathrm{i})(v_j(O_r) - \overline{\mathrm{V}}_\mathrm{j}) \quad (15)$$

The distance between 2 queries Qi and Qj is calculated through the distance metric represented in Equation 16. If the value of corr(Vi, Vj) is high, it indicates that there is signi cant overlapping in the corresponding result sets of Qi and Qj. Equation 16 ensures that, the query pairs which have signi cant overlapping in their result sets, are considered closer to each other in distance.

$$distance(\mathrm{V_i}, \mathrm{V_j}) = \frac{1}{corr(\mathrm{V_i}, \mathrm{V_j})} \quad (16)$$

E. Algorithm

The proposed Web object caching technique is outlined in Algorithm 1. The technique involves two stages. In the first stage, cohesive query clusters are created so that, intra cluster queries have significant result set overlapping in order to maximize space efficiency inside cache, p clusters are created for the k query feature vectors by using Cluster Create(V1,V2,…Vk). The functioning mechanism of refine-cluster(clusterp) is as follows: initially, p clusters are created randomly, and each cluster is associated with a centroid, which is calculated by using the corresponding query feature vectors. For each query, the distance between the query and other cluster centroids is calculated by using the distance metric represented in Equation 16. If the centroids of other clusters are nearer to the query compared to the existing centroid, then, the query is migrated to the nearest centroid represented cluster. The new cluster centroids for all p clusters are recalculated. The query cluster reorganization process is again repeated multiple times, until, cohesive clusters are formed.

The second stage is responsible for storing the selected queries and their corresponding result set inside the cache. The queries that are stored in the cache will only store a single copy of the non-unique result set Web objects in-order to maximize space efficiency. In this stage, the p groups/clusters are sorted in descending order of their score through sort group desc-score(G1,G2, …Gp) in-order to yield the sorted group set [Gs1, Gs2, ….Gsp]. The group size of all the groups in the sorted set is calculated through calculate-group-size(G^s_1,G^s_2,…G^s_p). The stage 2 has totally three different cases.

The first case is when size(Gs1) > size(C). This implies that, all the queries and their corresponding result set of Gs1 does not fit into the cache. Some of the queries which have low score need to be dropped in order to fit Gs1 inside the cache. The queries of Gs1 are sorted in increasing order of their scores through sort query inc score(Gs1), and a pivot is created for the first query in the sorted list through pivot(Gs1). Each query pointed by the pivot is extracted through extract query(pivot(Gs1)), and pivot is incremented through pivot increment(pivot(Gs1)), and this process continues until Gs1 fits into the cache. Finally, Gs1 is stored in the cache through store-cache(Gs1).

In the second case, size(Gs1) < size(C), which means that the result set size of Gs1 is lesser than the cache size. This implies that, there is a scope to store other high scoring groups either completely or partially in the cache. The empty list LG is created through create empty list(). A pivot indicated by group pivot is created for the first group in the sorted group list through create group pivot([Gs1; Gs2; ::::Gsp]). The pointed group by group pivot is added to LG, and group pivot is incremented through pivot increment(group pivot), and this process continues until size(LG) size(C). The last group of LG indicated by Gt is selected through select group(LG, last), and its queries are sorted in increasing order of their scores through sort queries inc score(Gt) and a pivot is created to point the first query through pivot(Gt) . The query pointed by the pivot is extracted through extract query(pivot(Gt)), the pivot is incremented through pivot increment(Gt), and this process continues until size(LG) size(C). Finally, the list LG is stored in the cache through store cache(LG).

In the third case, size(Gs1) = size(C). Here, Gs1 is directly stored in the cache through store cache(Gs1).

The Theorem 1 provides a concrete correctness justification of the proposed Web object caching technique.

Theorem 1. Algorithm Correctness

The proposed Web object caching technique in Algorithm 1 provides high cache hit ratio with high probability.

Proof: It is evident that, the queries which are placed in the cache by Algorithm 1 usually have high scores. Suppose, Qi is placed in the cache and Qj is left out from the cache, and score(Qi) > score(Qj). By using the property of Poisson distribution, P(f reqt(Qj) f reqt(Qi) 0) e (p i p j)2 , since, i > j which implies that, in the future MTI, the probability that, the frequency of Qj being submitted indicated by f reqt(Qj) is greater than the frequency of Qi being submitted indicated by f reqt(Qi) is extremely low. Hence, the queries which are submitted with high frequency in the future have a higher chance to be located

inside the cache.

Algorithm 1 Web Object Caching Technique

```
[Stage 1: Query Clustering]
cluster_p = Cluster_Create(V_1, V_2, ....V_k)
refine_cluster(cluster_p)
[Stage 2: Cache storage]
[G_1^s, G_2^s, ....G_p^s] = sort_group_dec_score(G_1, G_2, ....G_p)
calculate_group_size(G_1^s, G_2^s, ....G_p^s)
[Case 1]
if size(G_1^s) > size(C) then
   pivot(G_1^s) = sort_query_inc_score(G_1^s)
   while size(G_1^s) > size(C) do
      extract_query(pivot(G_1^s))
      pivot_increment(pivot(G_1^s))
   end while
   store_cache(G_1^s)
end if
[Case 2]
if size(G_1^s) < size(C) then
   L_G = create_empty_list()
   group_pivot = create_group_pivot([G_1^s, G_2^s, ....G_p^s])
   while size(L_G) < size(C) do
      add_list(L_G, group_pivot)
      pivot_increment(group_pivot)
   end while
   G_t = select_group(L_G, last)
   pivot(G_t) = sort_queries_inc_score(G_t)
   while size(L_G) > size(C) do
      extract_query(pivot(G_t))
      pivot_increment(G_t)
   end while
   store_cache(L_G)
end if
[Case 3]
if size(G_1^s) = size(C) then
   store_cache(G_1^s)
end if
```

IV. Results and Discussions

A. Experimental Setup

Since, available Web object search engines cannot be accessed in open source format; a custom built Web object search engine is utilized to perform the required empirical analysis. The Wikipedia data set is utilized for building the object repository. The data set size is around 50GB. Around 105 Web objects were created using the Wikipedia data set. The cache size was varied between 1GB to 5GB. The queries used for executing the proposed Web object caching technique are termed as cache queries, and the queries used for empirical analysis are termed as test queries.

Totally, 10 users were asked to submit their queries on the custom built Web object search engine. The queries submitted by the users over a period of ten days were collected and used as cache queries. Each day was considered as MTI. After executing the proposed Web object caching technique, the users again submitted the test queries for three consecutive MTI. The number of cache queries varied between 500 to 1000. The number of test queries was around 100. The metric cache hit ratio is the portion of test queries which are located in the cache. The metric average execution time refers to the average time taken to execute all the test queries. The number of query groups used to perform caching is varied from p = 5 to p = 15.

For the ease of reference, the proposed Web object caching mechanism is termed as new-cache, which is compared against the contemporary technique [5], which is termed as old-cache. The old-cache does not perform query caching, but, it performs Web object caching. So, a test query is considered to be located in the cache, only if, the entire result set of the query is found in the cache.

B. Empirical Result and Discussions

The first experiment analyzes the performance of old-cache and new-cache with respect to cache hit ratio when the cache capacity is varied, and the empirical result is illustrated in Figure 2. The increase in cache capacity provides an opportunity to store more queries in the cache, and which results in better cache hit ratio. Since, old-cache only stores the Web objects instead of queries, and due to its primitive design; new-cache out performs old-cache. The result of the first experiment with respect to average execution time is illustrated in Figure 3. Due to superior cache hit ratio of new-cache, the average execution time also improves.

The second experiment analyzes the performance of old-cache and new-cache with respect to cache hit ratio when the number of cache queries is varied, and the result is illustrated in Figure 4. As the number of cache queries increase, the query distribution information becomes more precise, which in-turn helps in identifying queries which have high potential to be submitted in future. Hence, cache hit ratio improves with the increase in number of cache queries. The result of the second experiment with respect to average execution time is illustrated in Figure 5. The new-cache outperforms old-cache for the same reasons explained above.

The third experiment analyzes the performance of new-cache with respect to cache hit ratio when the parameter p is varied; the old-cache is ignored, because p has no significance, and the result is illustrated in Figure 6. Large values of p indicates more number of query groups, which exhibit high intra query and low inter query cohesiveness. Similarly, low values of p indicate less number of query groups which have decent intra query cohesiveness. For higher values of p more number of query groups get opportunity to be considered for being stored in the cache, and since, inter group cohesiveness is limited, only few queries get an opportunity to be stored in the cache, because of minimal overlapping between the result sets of inter group queries. Hence, the cache hit ratio reduces. The result of the third experiment with respect to average execution time is illustrated in Figure 7.

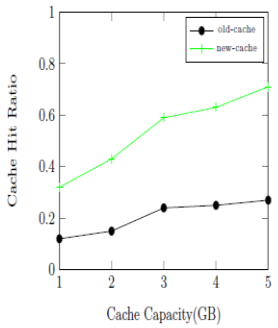

Fig. 2. Cache Hit vs Cache Capacity

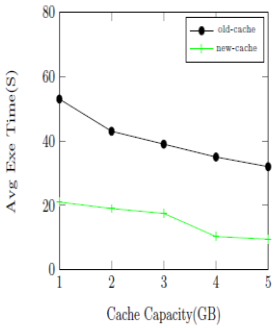

Fig. 3. Cache Capacity vs Exe Time

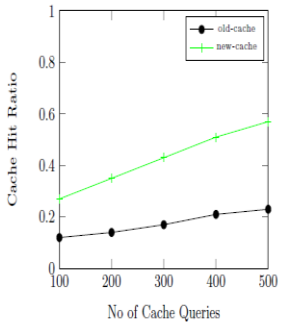

Fig. 4. Cache Hit vs No of Cache Queries

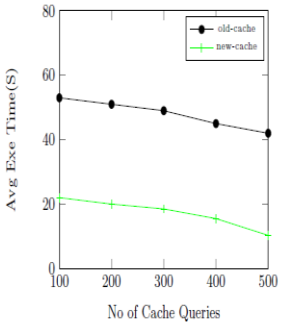

Fig. 5. No of Cache Queries vs Exe Time

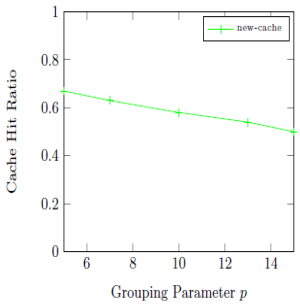

Fig. 6. Cache Hit vs p

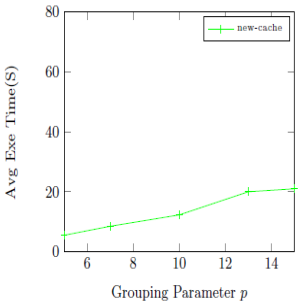

Fig. 7. p vs Exe Time

V. CONCLUSION

In this work, the importance of caching in Web object search engines was out-lined. The limitation of the contemporary caching technique [5] was highlighted. A new Web object caching mechanism based on probabilistic modeling was presented. The proposed caching technique achieves dual goal of storing queries which have high potential for being submitted in future, and storing more number of queries which have significant overlapped result set. The empirical results demonstrate the improved effectiveness of the proposed technique over the contemporary technique [5] with respect to cache hit ratio and execution time.

In future, the effectiveness of the proposed Web object caching technique; when applied to conventional IR systems need to be analyzed.

REFERENCES

1. Lu, Zhihong McKinley, Kathryn S.: Partial Collection Replication Versus Caching for Information Retrieval Systems. Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2000).
2. Weizhe Zhang, Hui He, Jianwei Ye.: A Two-Level Cache for Distributed Information Retrieval in Search Engines. The Scientific World Journal (2013).
3. A. Anand, X. H. Peng, R. Haywood.: E cient information retrieval via proxy caching mechanisms within a lossy environment. International Conference for Internet Technology and Secured Transactions (2009).
4. Lee, Ryong, Goshima, K, Kambayashi, Y, Takakura, H.: Caching Schema for Mobile Web information Retrieval. Technical Report (2017).
5. https://technet.microsoft.com/en-us/library/cc995252.aspx
6. Deng Cai, Xiaofei He, Ji-Rong Wen, Wei-Ying Ma.: Block Level Link Analysis. In Proceedings of SIGIR (2004).
7. Jaap Kamps, Maarten de Rijke, Borkur Sigurbjornsson Length normalization in XML retrieval. In proceedings of the SIGIR (2004).
8. Charles L A Clarke.: Controlling Overlap in Content oriented XML Retrieval. In Proceedings of the SIGIR (2005).
9. J.P Callan.: Distributed information retrieval. In Advances in Information Retrieval Recent research from the center for Intelligent Information Retrieval (2000).
10. L.Gravano, H.Garcia-Molina.: Generalizing gloss to vector space databases and broker hierarchies. In proceedings of the International Conference on Very Large Databases (1995).
11. Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, Wei-Ying Ma.: Web Object Retrieval. WWW (2007).
12. Dingming Wu, Gao Cong, Christian S.Jensen.: A frame work for e cient spatial Web object retrieval. The VLDB journal (2012).
13. Taro Tezuka, Hiroyuki Kondo, Kastumi Tanaka.: Estimation of Geographic Rele-vance for Web objects Using Probabilistic Models. Springer-Verlag Berlin Heidelberg (2008).

AUTHORS PROFILE

T S Bhagavath Singh working as Associate Professor in dept. of Information Science & Engineering with overall 20 years of teaching experience.

Dr. S Chitra M.E. Ph.D., working as Principal with overall 25 years of experience 5 students have been awarded Ph.D., and 5 Students are pursuing Ph.D.