Efficient Web Object Caching Through Query Correlation Approach

T. S. Bhagavath Singh and S. Chitra

Abstract Web object search engines are a new breed of Web Information Retrieval systems, wherein, the focus is to provide relevant results, such that, the result set has a 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 and stadiums. 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 query clustering 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

1 Introduction

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

T. S. Bhagavath Singh
Department of CSE, Anna University, Chennai, India
e-mail: rnsit.tpo@gmail.com

S. Chitra (✉)
Er. Perumal Manimekalai College of Engineering, Hosur, India
e-mail: schitra3@gmail.com

G. R. Kadambi et al. (eds.), *Emerging Trends in Photonics, Signal Processing and Communication Engineering*, Lecture Notes in Electrical Engineering 649,
https://doi.org/10.1007/978-981-15-3477-5_10

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.

Overview on IR Caching

Generally, Caching refers to storage of query accessed objects—which have high potential to be retrieved large number of times by the future queries—inside fast access storage system. 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 must minimize cache miss ratio.

Reserving component have become a significant part of numerous contemporary IR framework [1–4], in order to improve their reaction time. Most of these caching mechanisms store queries which are submitted with high frequency and their corresponding result set.

Contributions

In this work, the following contributions are made:

1. A new distributed Web object caching mechanism is proposed; wherein, queries which have high potential to be submitted in future and their corresponding result set are stored in query cache. The cache queries are selected through the aid of a scoring function. These selected queries are grouped based on their similarity w.r.t result sets. The queries in each group are likely to have good number of overlapping results. This grouping is achieved through correlation coefficient function. Each group is again scored w.r.t the number of high potential queries present in it. The highest ranked groups are selected to be placed in the cache storage.
2. The proposed caching mechanism is implemented in a custom-built Web object search engine. The object repository is created by using real-world Wikipedia data set. The empirical results demonstrate orders of magnitude improvement in query response time and superior cache hit ratio when compared with the contemporary caching technique [5].

2 Query Correlation Oriented Web Object Caching Technique

Architecture

The architecture of the proposed Web object search engine is illustrated in Fig. 1.

The query processing component is responsible for performing pre-processing on the user query and to identify the different object data sources, which can provide

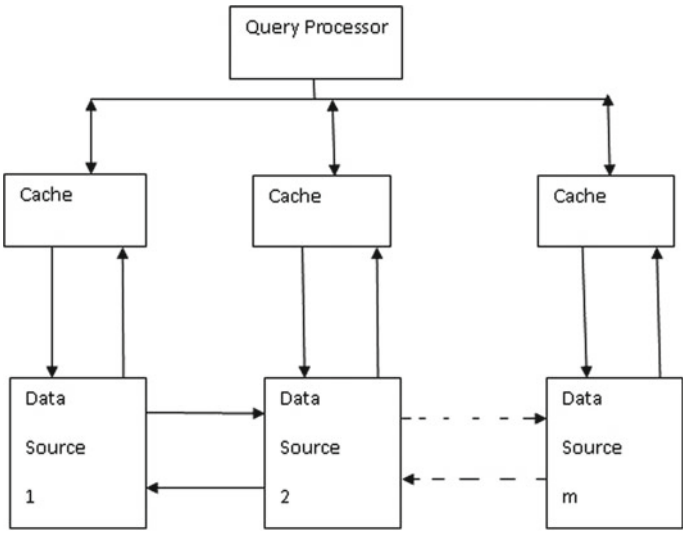

Fig. 1 Web object search engine architecture

the required results. Distributed architecture is utilized for achieving the required scalability in Big Data environments. Each object data source also contains a cache storage system.

Since the result set of many queries might contain Web objects which belong to different object data sources, and due to the availability of excellent bandwidth facility in today's systems, it is assumed that the cache corresponding to a specific object data source can store Web objects which might belong to other object data sources.

The Web object search engine is composed of m object data sources indicated by $D_1, D_2, \ldots, D_m$. The cache system for the $D_i (1 \leq i \leq m)$ object data source is indicated by C_i. The storage capacity of C_i is indicated by $size(C_i)$. The cache capacity of the entire Web object search engine is indicated by $size(C)$, where C indicates the cache for the entire Web object search engine which includes all the individual caches of different Web object data sources, and $size(C)$ is represented in Eq. (1).

$$size(C) = \sum_{t=1}^{m} size(C_i) \tag{1}$$

Query Score Function

In this section, scoring function to decide the potential of a query to be submitted in future is presented. Let, there be k queries present in the query log, and these queries are indicated by $Q_1, Q_2, \ldots, Q_k$. The queries which are present in the query log for a defined time interval T are selected, in order to obtain the recent submitted query statistics. The score for the query $Q_j (1 \leq j \leq k)$ is represented in Eq. (2). Here,

$score(Q_j)$ indicates the score of Q_j, larger score indicates higher potential of Q_j being submitted in the future, $freq(Q_j)$ indicates the frequency Q_j in the extracted section of the query log and hot $time(Q_j)$ indicates the difference between T and the last time Q_j has appeared in the query log.

$$score(Q_j) = \frac{freq(Q_j)}{hot\_time(Q_j)} \tag{2}$$

Consider a query group G_b which contains $Q_{|G_b|}$ number of queries indicated by $Q_1, Q_2, \ldots Q_{|G_b|}$. The score for this group is represented in Eq. (3). Here, group $score(G_b)$ indicates the group score of G_b, and higher group score indicates that this group contains queries which have high potential to be submitted in future.

$$group\_score(G_b) = \sum_{j=1}^{|G_b|} score(Q_j \in G_b) \tag{3}$$

Query Feature Vector

In this section, the feature vector required for detecting query correlation is presented. Let, n Web objects be present in the object repository, which includes all the individual object data sources. These Web objects are identified as $O_1, O_2, \ldots, O_n$. The query feature vector for Q_j is represented in Eq. (4). Here, V_j indicates the query feature vector for Q_j and $v(O_i)(1 \leq i \leq n)$ is represented in Eq. (5).

$$V_j = \begin{bmatrix} v(O_1) \\ v(O_1) \\ \vdots \\ v(O_n) \end{bmatrix} \tag{4}$$

$$V(O_i) == \begin{cases} 1, & \text{if } O_i \text{belongs to the result set of } Q_j \\ 0, & \text{Otherwise} \end{cases} \tag{5}$$

The group feature vector represents a query group. The group feature vector of G_b is represented in Eq. (6). Here, it represents the logical or operator, wherein the components in a specific position for two vectors are subjected to the logical or operation, $group\_vector(G_b)$ indicates the group feature vector of G_b and $V_1, V_2, \ldots, V_{|G_b|}$ indicate the query feature vectors of all the queries $Q_1, Q_2, \ldots, Q_{|Gb|}$, which $\in G_b$.

$$group\_vector(G_b) = V_1||V_2||\ldots||V_{||G_b||} = \begin{bmatrix} V_G(O_1) \\ V_G(O_2) \\ \vdots \\ V_G(O_n) \end{bmatrix} \tag{6}$$

The size of $size(G_b)$ is represented in Eq. (7). Here, $size(O_i)(1 \leq i \leq n)$ indicates the size of Web object O_i and $size(G_b)$ indicates the size of G_b. The Eq. (7) indicates that if any Web object O_i is part of the result set of many queries that $\in G_b$, then only a single copy of O_i will be considered inside the group G_b.

$$size(G_b) = \sum_{t=1}^{n} v_G(O_i) \times size(O_i) \tag{7}$$

Query Correlation Function

In this section, the correlation function to detect similarity or distance between queries is presented. The correlation function for the queries Q_i and Q_j, where, $i\ f = j$ is represented in Eq. (8). Here, $corr(V_i, V_j)$ indicates the function value, $var(V_i)$ and $var(V_j)$ indicate the variance of the values present in V_i and V_j, respectively, and they are represented in Eqs. (9) and (10), respectively. Here, V_i and V_j indicate the sample mean of V_i and V_j, respectively, and which are represented in Eqs. 11 and 12, respectively. The function $covar(V_i, V_j)$ indicates the covariance of the values present in V_i and V_j, and it is represented in Eq. (13).

$$corr(V_j, V_j) = \frac{covar(V_i, V_j)}{\sqrt{var(V_i)var(V_j)}} \tag{8}$$

$$var(V_i) = \frac{1}{n-1} \sum_{r=1}^{n} (v(O_r) - \overline{v_i})^2 \tag{9}$$

$$var(V_j) = \frac{1}{n-1} \sum_{r=1}^{n} \left(v(O_r) - \overline{v_j}\right)^2 \tag{10}$$

$$\overline{V_i} = \frac{\sum_{r=1}^{n} v(O_r)}{n} \tag{11}$$

$$\overline{V_j} = \frac{\sum_{r=1}^{n} v(O_r)}{n} \tag{12}$$

$$covar(V_i, V_j) = \frac{1}{n-1} \sum_{r=1}^{n} (v(O_r) - \overline{v_i}) \left(v(O_r) - \overline{v_j}\right) \tag{13}$$

The distance between 2 queries Q_i and Q_j is calculated through the distance metric represented in Eq. 14. If the value of $corr(V_i, V_j)$ is high, it indicates that there is significant overlapping in the corresponding result sets of Q_i and Q_j. Equation 14 ensures that the query pairs which have significant overlapping in their result sets are considered closer to each other in distance.

$$distance(V_i, V_j) = \frac{1}{corr(v_i, v_j)} \tag{14}$$

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 $Cluster\_Create(V_1, V_2, \ldots, V_k)$. The functioning mechanism of refine $cluster(cluster_p)$ 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 Eq. 14. 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.

3 Results and Discussions

Experimental Setup

To perform empirical analysis on the proposed caching system, custom-built Web object search engine is utilized, because the available Web object search engines cannot be accessed in open source format. The Wikipedia data set is utilized for building the object repository. The size of the data set is around 50 GB. The number of test samples performed on the cache queries were around 100.

Web objects were created using the Wikipedia data set. The cache size was varied between 1 and 5 GB. The queries used for executing caching mechanism are termed as cache queries. The subset of cache queries is stored in the cache. The queries which are used for analyzing the performance of caching mechanism 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 3–5 days were collected and used as cache queries. After executing the caching mechanism, the users were asked to again submit queries, which were used as test queries.

The number of cache queries varied between 500 and 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

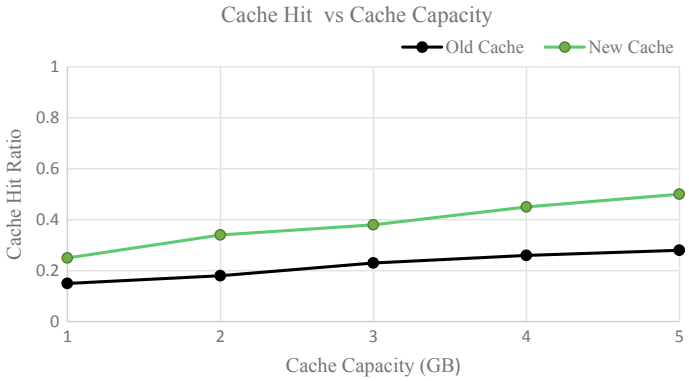

Fig. 2 Cache hit versus cache capacity

taken to execute all the test queries. The number of query groups used to perform caching is varied from $p = 5$ to $p = 15$.

The proposed caching mechanism is termed as new-cache for the ease of reference. This proposed caching mechanism is compared with 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.

Empirical Result Discussions

The first experiment analyzes the performance of old-cache and new-cache when the cache capacity is varied. The result w.r.t cache hit ratio is illustrated in Fig. 2. The increase in cache capacity provides an opportunity to store more queries and their corresponding result set in the cache, which results in better cache hit ratio. Since old-cache only stores the Web objects, and due to its primitive design, new-cache outperforms old-cache. The result w.r.t average execution time is illustrated in Fig. 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 when the number of cache queries is varied. The result with respect to cache hit ratio is illustrated in Fig. 4. As the number of cache queries increases, 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 with respect to average execution time is illustrated in Fig. 5. The new-cache outperforms old-cache for the same reasons explained before.

The third experiment analyzes the performance of new-cache when the parameter p is varied. The old-cache is ignored in this experiment, because the parameter p has no influence on it. The result with respect to cache hit ratio is illustrated in Fig. 6. Large values of parameter p indicates that more number of query groups, which

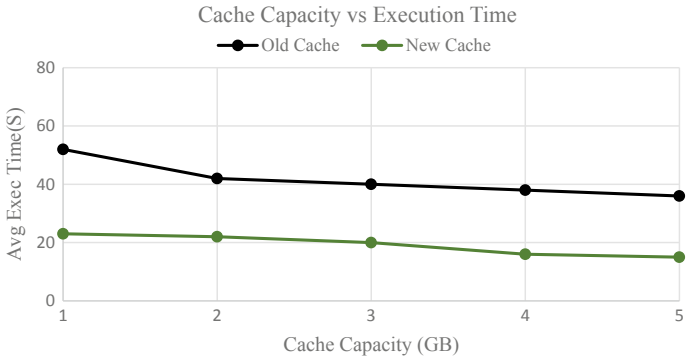

Fig. 3 Cache capacity versus exe time

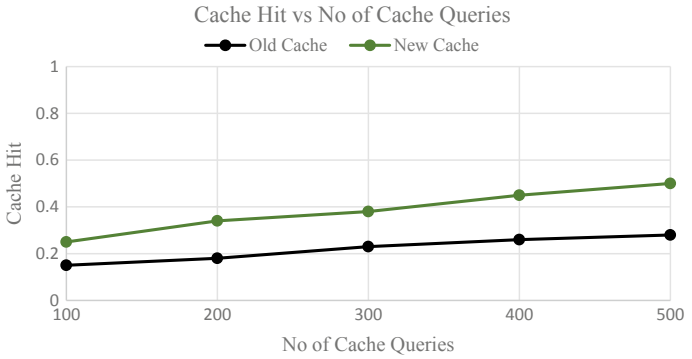

Fig. 4 Cache hit versus no of cache queries

Fig. 5 No of cache queries versus exe time

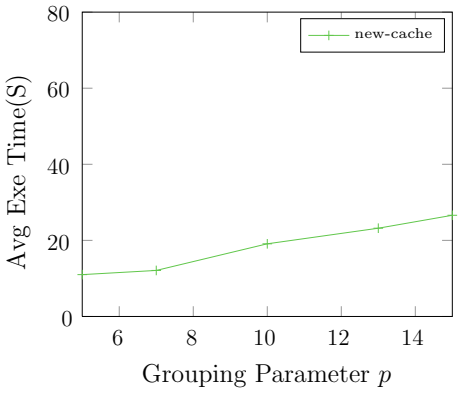

Fig. 6 Cache hit versus p

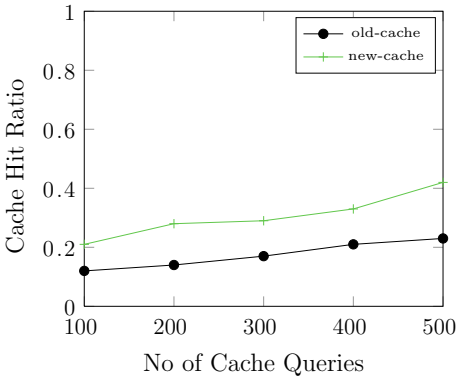

Fig. 7 p versus exe time

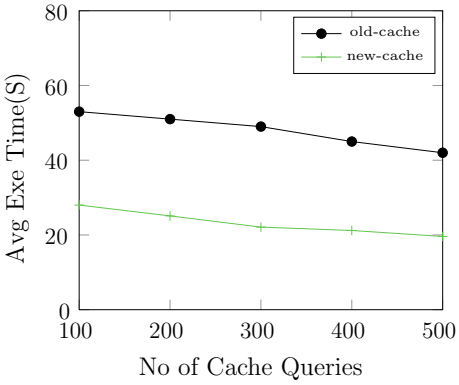

exhibit high intra query and low inter query cohesiveness, are created. Lower values of p indicate that small number of groups, which have good intra query cohesiveness, are created. For higher values of p, more number of query groups get opportunity to be considered for being stored in the cache. Since 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 with respect to average execution time is presented in Fig. 7.

4 Conclusion

In this work, the importance of caching in Web object search engines was described. The limitations of the contemporary caching technique [5] were highlighted. A new caching mechanism based on query correlation approach was presented. This proposed caching technique achieves dual goal of storing queries which have high potential in being submitted in future, and more number of queries which have significant

Algorithm 1 Web Object Caching Technique

[Stage 1: Query Clustering]
$cluster_p = Cluster\_Create(V_1, V_2, \ldots, V_k)$
$refine\_cluster(cluster_p)$
[Stage 2: Cache storage]
$[G_1^s, G_2^s, \ldots G_p^s] = sort\_group\_clec\_score(G_1, G_2, \ldots, G_p)$
$calculate\_group\_size(G_1^s, G_2^s, \ldots G_p^s)$
[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**
  $L_G = create\_empty\_list()$
  $group\_pivot = create\_group\_pivot([G_1^s, G_2^s, \ldots G_p^s])$
  while $size(L_G) < size(C)$ **do**
    $add\_list(L_G, group\_pivot)$
    $pivot\_incrernent(group\_pivot)$
  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**
    $extract\_query(pivot(G_t))$
    $pivot\_incrernent(G_t)$
  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

overlapped result set are selected for storing in cache. The empirical results demonstrate the improved effectiveness of the proposed technique over the contemporary technique [5] w.r.t cache hit ratio and execution time.

In future, dimensionality reduction techniques need to be explored for their effectiveness in designing Web object caching mechanism. The contemporary technique [5] has a different caching approach than the proposed caching mechanism, wherein, instead of queries and their result set, the Web objects are stored in the cache. New mechanisms which can improve the results for this caching approach also need to be explored.

References

1. Lu Z, McKinley KS (2000) Partial collection replication versus caching for information retrieval system. In: Proceedings of 23rd annual international ACM SIGIR conference on research and development in information retrieval
2. Zhang W, He H, Ye J (2013) A two-level cache for distributed information retrieval in search engines. Sci World J
3. Anand A, Peng XH, Haywood R (2009) Efficient information retrieval via proxy caching mechanisms within a lossy environment. In: International conference for internet technology and secured transactions
4. Lee R, Goshima K, Kambayashi Y, Takakura H (2017) Caching scheme for mobile web information retrieval. Technical report
5. http://techniet.microsoft.com/en-us/library/cc995252.aspx