Rough Web Caching

Sarina Sulaiman[1], Siti Mariyam Shamsuddin[1], and Ajith Abraham[2]

[1] Soft Computing Research Group, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Johor, Malaysia
{sarina,mariyam}@utm.my

[2] Centre for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology, Trondheim, Norway
ajith.abraham@ieee.org

Summary. The demand for Internet content rose dramatically in recent years. Servers became more and more powerful and the bandwidth of end user connections and backbones grew constantly during the last decade. Nevertheless users often experience poor performance when they access web sites or download files. Reasons for such problems are often performance problems, which occur directly on the servers (e.g. poor performance of server-side applications or during flash crowds) and problems concerning the network infrastructure (e.g. long geographical distances, network overloads, etc.). Web caching and prefetching have been recognized as the effective schemes to alleviate the service bottleneck and to minimize the user access latency and reduce the network traffic. In this chapter, we model the uncertainty in Web caching using the granularity of rough set (RS) and inductive learning. The proposed framework is illustrated using the trace-based experiments from Boston University Web trace data set.

1 Introduction

Good interactive response-time has long been known to be essential for user satisfaction and productivity [1, 2, 3]. This is also true for the Web [4, 5]. A widely-cited study from Zona Research [6] provides an evidence for the "eight second rule" in electronic commerce, *"if a Web site takes more than eight seconds to load, the user is much more likely to become frustrated and leave the site"*.

Lu *et al.*[7] has mentioned that most business organizations and government departments nowadays have developed and provided Internet based electronic services (e-services) that feature various intelligent functions. This form of e-services is commonly called *e-service intelligence* (ESI). ESI integrates intelligent technologies and methodologies into e-service systems for realizing intelligent Internet information searching, presentation, provision, recommendation, online system design, implementation, and assessment for Internet users. These intelligent technologies include machine learning, soft computing, intelligent languages, and data mining etc. ESI has been recently identified as a new direction for the future development stage of e-services. E-services offer great opportunities and challenges for many areas of services, such as government, education, tourism, commerce, marketing, finance, and logistics. They involve various online service providers, delivery systems and applications including e-government, e-learning, e-shopping, e-marketing, e-banking, and e-logistics.

A surprising fact is that many people tend to access the same piece of information repeatedly [7, 8] in any ESI. This could be weather related data, news, stock quotes, baseball scores, course notes, technical papers, exchange rate information and so on. If too many people attempt to access a Web site simultaneously, then they may experience problems in getting connected to the Web site. This is due to slow responses from the server as well as incapability of Web site in coping with the load.

An alternative way to tackle these problems is an implementation of Web caching in enhancing Web access [9, 8]. Web caching is beneficial to broad users including those who are relied on slow dial-up links as well as on faster broadband connections. The word caching refers to the process of saving data for future use. In other words, Web caching is the process of saving copies of content from the Web closer to the end user for quicker access. Web caching is a fairly new technology whose history is linked to that of the Web [10].

At the same time, Web prefetching is another well-known technique for reducing user web latency by preloading the web object that is not requested yet by the user [8, 9, 11, 12]. In other words, prefetching is a technique that downloads the probabilistic pages that are not requested by the user but could be requested again by the same user. Conventionally, there is some elapse time between two repeated requests by the same user. Prefetching usually performs the preloading operation within an elapse time and puts web objects into the local browser or proxy cache server to satisfy the next user's requests from its local cache.

However, the Web caching and prefetching technologies are the most popular software based solutions [11, 12]. Caching and prefetching can work individually or combined. The blending of caching and prefetching (called as pre-caching) enables doubling the performance compared to single caching [13]. These two techniques are very useful tools to reduce congestion, delays and latency problems. There are three most important features of web caching [14]:

- Caching that reduces network bandwidth usage
- Caching that also reduces user-perceived delays
- Caching that reduce loads on the original server

1.1 Problem in WWW Services

World Wide Web (WWW) has become the most ideal place for business and entertainment to enrich their presentation with interactive features. This has caused the evolution of Web growing and rising fast and drastically. Human interaction with objects or so called interactive features has leaded the Web to be more easily guided and capable to perform business task between distance places. These pages are linked and managed for certain purposes that perform as a Web application. These interactive Web pages consist of pages that are able to perform application logical task. The rising popularity of using Web applications in WWW causes tremendous demands on the Internet.

A key strategy for scaling the Internet to meet these increasing demands is to cache data near clients and thus improve access latency and reduce the network and server load [15, 16]. Mohamed *et. al* [17, 18, 19] has proposed an intelligent concept of *Smart Web Caching* with integrated modules of artificial neural networks (ANN), environment analysis and conventional caching procedure. The results are convincing in reducing the internet traffic flow and enhancing performances. However, implementing this integrated analyzer in Web caching environment causes highly computational cost [20, 17, 21] due to the complexity of the integrated process generation.

Caching is a technique used to store popular documents closer to the user. It uses algorithms to predict user's needs to specific documents and stores important documents. According to Curran and Duffy [22], caching can occur anywhere within a network, on the user's computer or mobile devices, at a server, or at an Internet Service Provider (ISP). Many companies employ web proxy caches to display frequently accessed pages to their employees, as such to reduce the bandwidth with lower costs [22, 23]. Web cache performance is directly proportional to the size of the client community [24, 22]. The bigger the client community, the greater the possibility of cached data being requested, hence, the better the cache's performance [22].

Moreover, caching a document can also cause other problems. Most documents on the Internet change over time as they are updated. Static and Dynamic Caching are two different technologies that widely used to reduce download time and congestion [20]. *Static Caching* stores the content of a web page which does not change. There is no need to request the same information repeatedly. This is an excellent approach to fight congestion. *Dynamic Caching* is slightly different. It determines whether the content of a page has been changed. If the contents have changed, it will store the updated version [23]. This unfortunately can lead to congestion and thus it is possibly not a very good approach as it does require verification on the source of the data prior to updating. If these two technologies are implemented simultaneously, then the latency and congestion can be diminished.

According to Davison [14] caching helps to bridge the performance gap between local activity and remote content. Caching assists improvement of Web performance by reducing the cost and end-user latency for Web access within a short term. However, in the long term, even as bandwidth costs continue to drop and higher end-user speeds become available; caching will continue to obtain benefits for the following reasons:

Bandwidth will always have some ***cost****.* The cost of bandwidth will never reach zero, even though the competition is increasing, the market is growing, and the economies of scale will reduce end-user costs. The cost of bandwidth at the core has stayed relatively stable, requiring ISPs to implement methods such as caching to stay competitive and reduce core bandwidth usage so that edge bandwidth costs can be low.

Nonuniform bandwidth and ***latencies*** *will persist.* Because of physical limitations such as environment and location as well as financial constraints, there will always be variations in bandwidth and latencies. Caching can help to smooth these effects.

Network ***distances*** *are increasing.* Firewalls, other proxies for security and privacy, and virtual private networks for telecommuters have increased the number of hops for contents delivery, hence slow Web response time.

Bandwidth ***demands*** *continue to increase.* The growth of user base, the popularity of high-bandwidth media, and user expectations of faster performance have guaranteed the exponential increase in demand for bandwidth.

Hot spots *in the Web will continue.* Intelligent load balancing can alleviate problems when high user demand for a site is predictable. However, a Web site's popularity can also appear as a result of current events, desirable content, or gossips. Distributed Web caching can help alleviate these "hot spots" resulting from flash traffic loads.

Communication ***costs*** *exceed computational costs.* Communication is likely to always be more expensive (to some extent) than computation. The use of memory caches are preferred because CPUs are much faster than main memory. Likewise, the cache mechanisms will prolong as both computer systems and network connectivity become faster.

Furthermore, caching is the most relevant technique to improve storage system, network, and device performance. In mobile environments, caching can contribute to a greater reduction in the constraint of utilization resources such as network bandwidth, power, and allow disconnected operation [29]. A lot of studies are focused on developing a better caching algorithm to improve the choice of item to replace, and simultaneously, building up techniques to model access behavior and prefetch data. From 1990's until today, researchers on caching have produced different caching policies to optimize a specific performance and to automate policy parameter tuning. Prior to this, administrator or programmer had to select a particular parameter to observe workload changes. However, an adaptive and self-optimizing caching algorithm offer another advantage when considered mobile environments, where users of mobile devices should not expect to tune their devices to response the workload changes [29]. The workload depends on the current position of the mobile node in relation to other nodes and stations, and also depends on the current location and context of the mobile user.

Caching is effectively for data with infrequent changes. Besides, caching data locally to mobile nodes helps the ability to retrieve data from a nearby node, rather than from a more distant base station [28]. By simply retrieving data using multiple short-range transmissions in wireless environments provides a reduction in overall energy consumed. Santhanakrishnan *et al.* [29] illustrated on the demand-based retrieval of the Web documents in the mobile Web. They proposed caching scheme; Universal Mobile Caching which performed the most basic and general form of caching algorithms and largely emphasize the impact of the adaptive policy. This scheme is suitable for managing object caches in structurally varying environments. Ari *et al.* [30] proposed Adaptive Caching using Multiple Experts (ACME), which the individual experts were full replacement algorithms, applied to virtual caches, and their performance was estimated based on the observed performance of the virtual caches. The term expert refers to any mechanism for offering an answer to the question. For cache replacement, the answer they seek is the identity of the object in the cache with the least likelihood of subsequent future access.

Contrast to a single algorithm, there are not so many research works on integrated schemes. Aiming at integrating caching and prefetching, Yang and Zhang [26] employed a prediction model, whereas Teng *et al.* [31] presented a new cache replacement algorithm, considering the impact of prefetching engine located at Web server and a few cache parameters. Kobayashi and Yu [32] discussed the performance model for mobile Web caching and prefetching and provided the estimate of the total average latency, hit ratio, cache capacity and wireless bandwidth required.

Prefetching is an intelligent technique used to reduce perceived congestion, and to predict the subsequent page or document to be accessed [24, 12]. For example, if a user is on a page with many links, the prefetching algorithm will predict that the user may want to view associated links within that page. The prefetcher will then appeal the predicted pages, and stores them until the actual request is employed. This approach will display the page significantly faster compared to the page request without prefetching. The only drawback is that if the user does not request the pages, the prefetching algorithm will still implement the prediction of the subsequent pages, thus causes the network to be congested [25, 26, 27, 28].

In addition, Web prefetching method evolves from prefetching top-10 popular pages [33] or hyperlinks [34] into prefetching by user's access patterns. Statistical prefetching algorithms [35] make use of Markov modeling, and establish a Markov graph based on user's access histories and make prefetching predictions based on the graph which needs to be updated continuously while accessing Web. Prefetching strategies in [25, 36] used data mining technique, to decide whether to prefetch or not according to the probability of the pages accessed recently. But it is possible that the prefetched pages are far away from the current page sequence so that the cache hit ratio may not benefit from prefetching.

Hence, Web prefetching strategy need to achieve a balance between network loads and performance gains. Some research studies have found that too aggressive prefetching will increase Web access latency, since more prefetching will lead to replacement of more cache items even including the pages that will be accessed in near future. Under the wireless environment, Yin and Cao [37] proposed to dynamically adjust the number of prefetching according to power consumption for mobile data dissemination.

Wu *et al.* [38] introduced a rule-based modular framework for building self-adaptive applications in mobile environments. They developed techniques that combine static and dynamic analysis to uncover phase structure and data access semantics of a rule program. The semantic information is used to facilitate intelligent caching and prefetching for conserving limited bandwidth and reducing rule processing cost. As well, Komninos and Dunlop [39] found that calendars can really provide information that can be used to prefetch useful Internet content for mobile users. While it is expected that such an approach cannot fulfill the whole of Internet content needs for a user, the work presented provided evidence to the extent to which a mobile cache can be populated with relevant documents that the user could find of interest. However, a foreseeable problem with the current system is that the current adaptation algorithm adjusts the system gradually, and not immediately, to the needs of a user. Thus, if a dramatic change of circumstances was to occur, or if a user was to require information from a very specific and known source, it is likely the system would fail to provide the necessary information.

2 Why Web Caching?

Web caching is the temporary storage of Web objects (such as HTML documents) for later retrieval. There are three significant advantages to Web caching: reduced bandwidth consumption (fewer requests and responses that need to go over the network), reduced server load (fewer requests for a server to handle), and reduced latency (since responses for cached requests are available immediately, and closer to the client being served). Together, they make the Web less expensive and better performing.

Caching can be performed by the client application, and is built in to most Web browsers. There are a number of products that extend or replace the built-in caches with systems that contain larger storage, more features, or better performance. In any case, these systems cache net objects from many servers but all for a single user.

Caching can also be utilized in the middle, between the client and the server as part of a proxy. Proxy caches are often located near network gateways to reduce the bandwidth required over expensive dedicated Internet connections. These systems serve many users (clients) with cached objects from many servers. In fact, much of the usefulness (reportedly up to 80% for some installations) is in caching objects requested by one client for later retrieval by another client. For even greater performance, many proxy caches are part of cache hierarchies, in which a cache can inquire of neighboring caches for a requested document to reduce the need to fetch the object directly.

Finally, caches can be placed directly in front of a particular server, to reduce the number of requests that the server must handle. Most proxy caches can be used in this fashion, but this form has a different name (reverse cache, inverse cache, or sometimes httpd accelerator) to reflect the fact that it caches objects for many clients but from (usually) only one server [21].

2.1 How Web Caching Works?

All caches have a set of rules that they use to determine when to serve an object from the cache, if it's available. Some of these rules are set in the protocols (HTTP 1.0 and 1.1), and some are set by the administrator of the cache (either the user of the browser cache, or the proxy administrator).

Generally speaking, these are the most common rules that are followed for a particular request [21]:

1. If the object's headers notify the cache not to keep the object, then it will do so. Simultaneously, if there is no validation, then most caches will mark that as uncacheable item.
2. If the object is authenticated or secured, then it will not be cached.
3. A cached object is considered *fresh* (that is, able to be sent to a client without checking with the origin server) if:
   - It has an expiry time or other age-controlling directive set, and is still within the fresh period.
   - If a browser cache has already seen the object, and has been set to check once a session.
   - If a proxy cache has seen the object recently, and it was modified relatively long ago. Fresh documents are served directly from the cache, without checking with the origin server.
4. If an object is stale, the origin server will be executed to *validate* the object, or notify the cache whether the existing copy is still good.

Mutually **freshness** and **validation** are the most important mechanisms that make cache works with content. A fresh object will be available instantly from the cache, while a validated object will avoid sending the entire object all over again if it has not been changed.

3 Performance Measurement for Web Optimization

Performance measurement of Web caching is needed to establish the efficiency of a Web caching solution [9, 17, 32]. Some performance benchmarks or standards are required for a particular Web caching solution to be evaluated. Such benchmarks may assist in choosing the most suitable Web caching solution for the problem we encounter. In this situation, a possibility of a particular structure will beneficial for certain applications while other applications may require some other substitutes.

Some organizations may choose for proxy based caching solutions. They may try to overcome the problem of configuration Web browsers by forcing the use of browsers that provide auto-configuration. For massive organizations, network components such as routers and switches [9, 10] might be considered; otherwise, transparent caching can be employed. Some organizations may prefer highly scalable solutions for anticipating future needs. Besides, organizations which Web sites contain highly dynamic content might occupy Active Cache [41] or possibly will utilize Web server accelerators. Obviously, the subject of measurement of performance is controlled not just to find the competence of a given Web caching solution but also to cover evaluation of the performance of cache consistency protocols, cache replacement algorithms, the role of fundamental protocols such as HTTP and TCP and others.

3.1 Parameters for Measuring Web Performance

Several metrics are commonly used when evaluating Web caching policies [41]. These include [42]:

1. Hit rate is generally a percentage ratio of documents obtained by using the caching mechanism and total documents requested. If measurement focuses on byte transfer efficiency, then weighted hit rate is a better performance measurement [43].
2. Bandwidth utilization is an efficiency metric measurement. The reduction bandwidth consumption shows that the cache is better.
3. Response time/access time –response time is the time taken for a user to get a document.

The are various parameters such as user access patterns, cache removal policy, cache size and document size that can significantly affect cache performance. Other common metrics that are used to quantify the performance of Web caching solutions proposed by Mohamed [17] include hit ratio, byte hit ratio, response time, bandwidth saved, script size and current CPU usage.

Performance of Web caching solutions may be quantified by measuring parameters as follows [9]:

1. price
2. throughput (e.g. the number of HTTP requests per second generated by users, the rate at which a product delivers cache hits etc.)
3. cache hit ratio (the ratio of the number of requests met in the cache to the total number of requests)
4. byte hit ratio (the fraction of the number of bytes served by the cache divided by the total number of bytes sent to its clients)
5. the number of minutes until the first cache hit/miss after a breakdown
6. the cache age (the time after which the cache become full)
7. hit ratio/price (e.g. hits/second per thousand dollars)
8. downtime (e.g. time to recover from power outrages or cache failures)

Techniques for measuring the efficiency and usefulness of Web caching solutions have been evolving slowly since this field is relatively a new discipline; the theory of Web Caching has advanced much faster than practice [9].

Despite quantifying the performance of caching clarifications, other aspects such as client side latencies, server side latencies, aborted requests, DNS lookup latencies, cookies, different popularity characteristics among servers, the type of content, network packet losses should not be disregarded since there are some parameters are interrelated. For illustration, hit ratio is affected by inadequate disk space in a cache server, and these lacking in the object placement/replacement policies can cause the network to be overloaded. Hence, by maximizing a single parameter alone may not be adequate [9].

4 Uncertainty in Web Caching

Uncertainty, as well as evolution, is a part of nature. When humans describe complex environments, they use linguistic descriptors of cognized real-world circumstances that are often not precise, but rather "fuzzy". The theory of fuzzy sets [44] provides an effective method of describing the behavior of a system, which is too complex to be handling with the classical precise mathematical analysis. The theory of rough sets [61] emerged as another mathematical approach for dealing with uncertainty that arises from inexact, noisy or incomplete information. Fuzzy set theory assumes that the membership of the objects in some set is defined as a degree ranging over the interval [0,1]. Rough Set Theory (RST) focuses on the ambiguity caused by the limited distinction between objects in a given domain.

Uncertainty occurs in many real-life problems. It can cause the information used for problem solving being unavailable, incomplete, imprecise, unreliable, contradictory, and changing [46]. In computerized system, uncertainty is frequently managed by using quantitative approaches that are computationally intensive. For example, a binary that processes 'TRUE or FALSE', or 'YES' or 'NO' type of decisions, is likely to arrive at a conclusion or a solution faster than one that needs to handle uncertainty.

Organizing uncertainty is a big challenge for knowledge-processing systems [46]. In some problems, uncertainty can possibly be neglected, though at the risk of compromising the performance of a decision support system. However, in most cases, the management of uncertainty becomes necessary because of critical system requirements or more complete rules are needed. In these cases, eliminating inconsistent or incomplete information when extracting knowledge from an information system may introduce inaccurate or even false results, especially when the available source information is limited. Ordinarily, the nature of uncertainty comes from the following three sources: incomplete data, inconsistent data, and noisy data.

Thus, in a proxy cache, the superfluous of logs dataset with the huge number of records, the frequency of errors (incomplete data), and the diversity of log formats (inconsistent data) [10] will ground the practical challenges to analyze it either to cache or not cache objects in the popular documents. Table 1 depicts the sample of Web log data from Boston University Web Trace [47].

4.1 How Rough Sets Boost Up Web Caching Performance?

Another approach to represent uncertainty is using Rough Set (RS). RS are based on equivalence relations and set approximations, and the algorithms for computing RS properties are combinatorial in nature. The main advantages of RST are as follows [48]:

- It does not need any preliminary or additional information about data;
- It is easy to handle mathematically;
- Its algorithms are relatively simple.

Table 1. Sample Web log data

```
bugs 791131220 682449 “http://cs-www.bu.edu/” 2009 0.518815
bugs 791131221 620556 “http://cs-www.bu.edu/lib/pics/bu-logo.gif” 1805
0.320793
bugs 791131222 312837 “http://cs-www.bu.edu/lib/pics/bu-label.gif” 717
0.268006
bugs 791131266 55484 “http://cs-www.bu.edu/courses/Home.html” 3279
0.515020
bugs 791131266 676413 “http://cs-www.bu.edu/lib/pics/bu-logo.gif” 0 0.0
bugs 791131266 678045 “http://cs-www.bu.edu/lib/pics/bu-label.gif” 0 0.0
bugs 791131291 183914 “http://cs-www.bu.edu/students/grads/tahir/CS111/”
738 0.292915
bugs 791131303 477482 “http://cs-www.bu.edu/students/grads/tahir/CS111/
hw2.ps” 41374 0.319514
bugs 791131413 265831 “http://cs-www.bu.edu/students/grads/tahir/CS111/if-
stat.ps” 10202 0.380549
bunsen 791477692 218136 “http://cs-www.bu.edu/” 2087 0.509628
bunsen 791477693 134805 “http://cs-www.bu.edu/lib/pics/bu-logo.gif” 1803
0.286981
bunsen 791477693 819743 “http://cs-www.bu.edu/lib/pics/bu-label.gif” 715
0.355871
bunsen 791477719 107934 “http://cs-www.bu.edu/techreports/Home.html” 960
0.335809
bunsen 791477719 518262 “http://cs-www.bu.edu/lib/pics/bu-logo.gif” 0 0.0
bunsen 791477719 520770 “http://cs-www.bu.edu/lib/pics/bu-label.gif” 0 0.0
```

Wakaki *et al.* [48] used the combination of the RS-aided feature selection method and the support vector machine with the linear kernel in classifying Web pages into multiple categories. The proposed method gave acceptable accuracy and high dimensionality reduction without prior searching of better feature selection. Liang *et al.* [49] used RS and RS based inductive learning to assist students and instructors with WebCT learning. Decision rules were obtained using RS based inductive learning to give the reasons for the student failure. Consequently, RS based WebCT Learning improves the state-of-the-art of Web learning by providing virtual student/teacher feedback and making the WebCT system much more powerful.

Ngo and Nguyen [50] proposed an approach to search results clustering based on tolerance RS model following the work on document clustering. The application of tolerance RS model in document clustering was proposed as a way to enrich document and cluster representation to increase clustering performance. Furthermore, Chimphlee *et al.* [51] present a RS clustering to cluster web transactions from web access logs and using Markov model for next access prediction. Users can effectively mine web log records to discover and predict access patterns while using this approach. They perform experiments using real web trace logs collected from www.dusit.ac.th servers. In order to improve its prediction ration, the model includes a rough sets scheme in which search similarity measure to compute the similarity between two sequences using upper approximation.

In [52], the authors employed RS based learning program for predicting the web usage. In their approach, web usage patterns are represented as rules generated by the inductive learning program, BLEM2. Inputs to BLEM2 are clusters generated by a hierarchical clustering algorithm that are applied to preprocess web log records. Their empirical results showed that the prediction accuracy of rules induced by the learning program is better than a centroid-based method, and the learning program can generate shorter cluster descriptions.

In general, the basic problems in data analysis that can be undertaken by using RS approach is as follows [46]:

- Characterization of a set of objects in terms of attribute values;
- Finding the dependencies (total or partial) between attributes;
- Reduction of superfluous attributes (data);
- Finding the most significant attributes;
- Generation of decision rules.

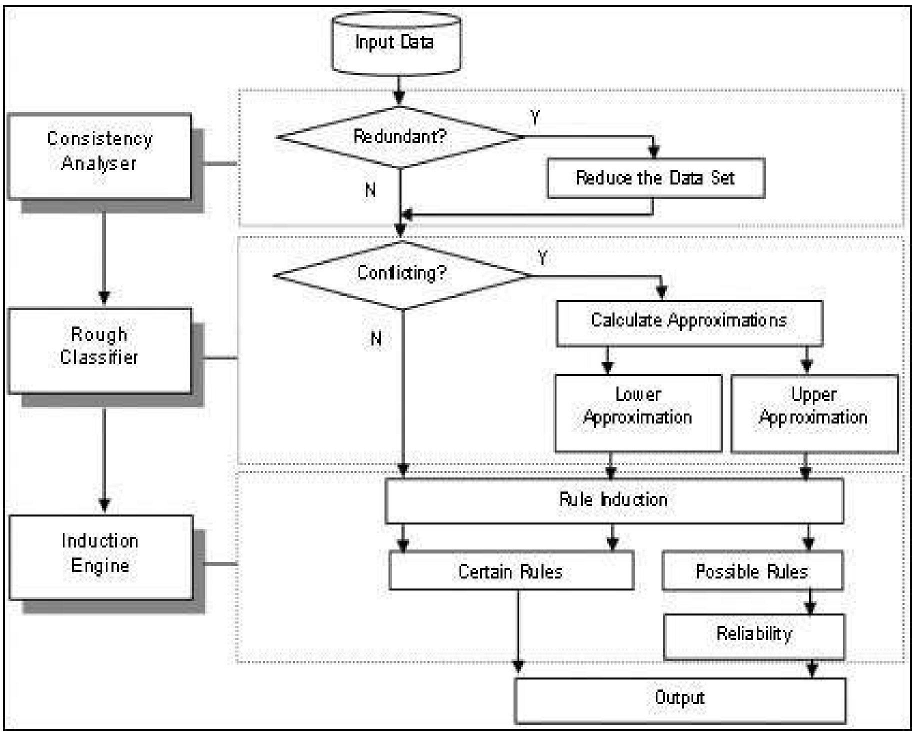

Fig. 1. Framework of the RClass System [46]

4.2 A Framework of Rough Sets

The RClass system integrates RST with an ID3-like learning algorithm [46] as shown in Figure 1. It includes three main modules; a consistency analyzer, a rough classifier and an induction engine. The consistency analyzer analyses the training data and performs two tasks; elimination of redundant data items, and identification of conflicting training data. The rough classifier has two approximators; the upper approximator and the lower approximator. The rough classifier is employed to treat inconsistent training data. The induction engine module has an ID3-like learning algorithm based on the minimum-entropy principle. The concept of entropy is used to measure how informative an attribute is.

5 Rough Sets and Inductive Learning

Rough Set Theory [53] was introduced by Zdzislaw Pawlak as a tool to solve problems with ambiguity and uncertainty [46]. Typically, data to be analyzed consists of a set of *objects* whose properties can be described by multi-valued *attributes.* The objects are described by the data that can be represented by a structure called the *information system* (S) [54]. An information system can be viewed as information table with its rows and columns consequent to objects and attributes.

Given a set E of examples described by an information table T, we classify objects in two different ways: by a subset C of the condition attributes and by a decision attribute D in the information table to find equivalence classes called indiscernibility classes $\Omega = \{\Omega_1, ..., \Omega_n\}$ [55]. Objects within a given indiscernibility class are indistinguishable from each other on the basis of those attribute values. Each equivalence class based on the decision attribute defines a concept. We use $Des(\Omega_i)$ [49] to denote the description, i.e., the set of attribute values, of the equivalence class Ω_i. RS theory allows a concept to be described in terms of a pair of sets, lower approximation and upper approximation of the class. Let Y be a concept. The lower approximation $\underline{Y}$ and the upper approximation $\overline{Y}$ of Y are defined as [49]:

$$\underline{Y} = \{e \in E | e \in \Omega_i and X_i \subseteq Y\} \tag{1}$$

$$\overline{Y} = \{e \in E | e \in \Omega_i and X_i \cap Y = \emptyset\} \tag{2}$$

Lower approximation is the intersection of all those elementary sets that are contained by Y and upper approximation is the union of elementary sets that are contained by Y.

Inductive Learning is a well-known area in artificial intelligence. It is used to model the knowledge of human experts by using a carefully chosen sample of expert decisions and inferring decision rules automatically, independent of the subject of interest [56]. RS based Inductive Learning uses RS theory to find general decision rules [57, 58]. These two techniques are nearness to determine the relationship between the set of attributes and the concept.

5.1 Rough Set Granularity in Web Caching

In our research, BU Web trace dataset from Oceans Research Group at Boston University are used [47]. We considered 20 sample objects only, i.e., January 1995 records. In our previous research, we used the same dataset with implementation of RS [59] and integration of Neurocomputing and Particle Swarm Optimization (PSO) algorithm [60] to optimize the Web caching performance. Three conditional attributes are taken into consideration; request time (*Timestamp, TS*) in seconds and microseconds, a current CPU usage (*Sizedocument, SD*) in bytes and response time (*Objectretrievaltime, RT*) in seconds. Consequently, a cache, CA is chosen as a decision for the information table; 1 for cache and 0 for not cache. Decision rules are obtained using RS based Inductive Learning [57] for

Table 2. Sample of log files dataset information table

Object	**Attributes**			**Decision**
	TS	***SD***	***RT***	***CA***
S_1	790358517	367	0.436018	0
S_2	790358517	514	0.416329	0
S_3	790358520	297	0.572204	0
S_4	790358527	0	0	1
S_5	790358529	0	0	1
S_6	790358530	0	0	1
S_7	790358530	0	0	1
S_8	790358538	14051	0.685318	0
S_9	790362535	1935	1.021313	0
S_{10}	790362536	1804	0.284184	0
S_{11}	790362537	716	0.65038	0
S_{12}	790363268	1935	0.76284	0
S_{13}	790363270	716	1.050344	0
S_{14}	790363270	1804	0.447391	0
S_{15}	790363329	1935	0.553885	0
S_{16}	790363330	716	0.331864	0
S_{17}	790363330	1804	0.342798	0
S_{18}	790363700	0	0	1
S_{19}	790363700	0	0	1
S_{20}	790363700	1136	0.428784	0

Web caching. Table 2 depicts the structure of the study: 20 objects, 3 attributes, and a decision.

Detailed description and analysis are given in Table 3. The domain E and two concepts Y_{cache} and $Y_{notcache}$ from the decision attribute (CA) are obtained as follows:

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}, e_{17}, e_{18}, e_{19}, e_{20}\}$
$Y_{cache} = \{e_4, e_5, e_6, e_{17}\}$
$Y_{notcache} = \{e_1, e_2, e_3, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}, e_{18}\}$

Initially we find the indiscernibility classes based on TS that are $\{e_1, e_2\}$, $\{e_{12}, e_{13}\}$, $\{e_{15}, e_{16}\}$,$\{e_{17}, e_{18}\}$and$\{e_3\}$, $\{e_4\}$,$\{e_5\}$,$\{e_6\}$,$\{e_7\}$, $\{e_8\}$,$\{e_9\}$,$\{e_{10}\}$, $\{e_{11}\}$, $\{e_{14}\}$.

The discriminant index of a concept Y is defined using the following formula:

$$\alpha_{C_i}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| \qquad (3)$$

Consequently, the discriminant index of TS is $\alpha_{C1}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| =$ 1-(9-0)/20 = 0.55 determines the effectiveness of the singleton set of attributes consisting of TS in specifying the membership in Y (the cache concept). Subsequently, the indiscernibility classes of SD is conducted and the results are $\{e_4, e_5, e_6, e_{17}\}$,$\{e_{10}, e_{12}, e_{15}\}$, $\{e_9, e_{13}, e_{16}\}$,$\{e_8, e_{11}, e_{14}\}$ and $\{e_1\}$,$\{e_2\}$,$\{e_3\}$,$\{e_7\}$, $\{e_{18}\}$.

The lower approximation is illustrated as
$\underline{Y} = \cup_{\Omega_i \subseteq Y} \Omega_i = \{e_1\},\{e_2\},\{e_3\}, \{e_7\},\{e_{18}\}$. The upper approximation is given as $\overline{Y} = \cup_{\Omega_i \cap Y \neq \emptyset} \Omega_i = \{e_4, e_5, e_6, e_{17}, e_{10}, e_{12}, e_{15}, e_9, e_{13}, e_{16}, e_8, e_{11}, e_{14}\}$. Hence, the discriminant index of SD is $\alpha_{C2}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (15 - 5)/20 = 0.5$.

The indiscernibility classes based on RT are $\{e_4, e_5, e_6, e_{17}\}$ and $\{e_1\}$,$\{e_2\}$, $\{e_3\}$,$\{e_7\}$,$\{e_8\}$,$\{e_9\}$,$\{e_{10}\}$,$\{e_{11}\}$,$\{e_{12}\}$,$\{e_{13}\}$,$\{e_{14}\}$,$\{e_{15}\}$,$\{e_{16}\}$,$\{e_{18}\}$. The lower approximation is given as $\underline{Y} = \cup_{\Omega_i \subseteq Y} \Omega_i = \emptyset$. The upper approximation is $\overline{Y} = \cup_{\Omega_i \cap Y \neq \emptyset} \Omega_i = \{e_4, e_5, e_6, e_{17}\}$. The discriminant index of RT is $\alpha_{C3}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| =$ 1 - (6 - 0)/20 = 0.7.

By comparing the discriminant indices of all attributes, we identify that the discriminant index of RT has the highest value, $\alpha_{Condition3}(Y) = 0.7$. This value determines better membership in Y. Hence, the first rule is obtained as:

$$\boldsymbol{R_1 : \{Objectretrievaltime = 0\} \Rightarrow \{Cache = 1\}}$$

Since RT is the most important condition attribute, we merge this condition attribute with other condition attributes to produce a new domain and to execute new rules (refer to Table 3).

To discover the new domain, initially, the following equation is used to remove unnecessary elements. $(E - \overline{Y}) \cup (\underline{Y}) = \{e_1, e_2, e_3, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}\} \cup \emptyset$. The new element set are given as, $(E - [(E - \overline{Y}) \cup (\underline{Y})] = (E - \{e_1, e_2, e_3, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}\} \cup \emptyset) = \{e_4, e_5, e_6, e_{17}\}$

Table 3. Collapsed log files dataset information table

Object	**Attributes**			**Decision**	**Total**
	TS	***SD***	***RT***	***CA***	
e_1	790358517	367	0.436018	0	1
e_2	790358517	514	0.416329	0	1
e_3	790358520	297	0.572204	0	1
e_4	790358527	0	0	1	1
e_5	790358529	0	0	1	1
e_6	790358530	0	0	1	2
e_7	790358538	14051	0.685318	0	1
e_8	790362535	1935	1.021313	0	1
e_9	790362536	1804	0.284184	0	1
e_{10}	790362537	716	0.65038	0	1
e_{11}	790363268	1935	0.76284	0	1
e_{12}	790363270	716	1.050344	0	1
e_{13}	790363270	1804	0.447391	0	1
e_{14}	790363329	1935	0.553885	0	1
e_{15}	790363330	716	0.331864	0	1
e_{16}	790363330	1804	0.342798	0	1
e_{17}	790363700	0	0	1	2
e_{18}	790363700	1136	0.428784	0	1

Table 4. Horizontal selection of collapsed table

Object	**Attributes**			**Decision**	**Total**
	TS	***SD***	***RT***	***CA***	
e_4	790358527	0	0	1	1
e_5	790358529	0	0	1	1
e_6	790358530	0	0	1	2
e_{17}	790363700	0	0	1	2

Table 5. Further horizontally collapsed reduction table

Object	Attributes	Decision	Total
	TS	***CA***	
e_4	790358527	1	1
e_5	790358529	1	1
e_6	790358530	1	2
e_{17}	790363700	1	2

Subsequently, the horizontal selection of the collapsed information table is obtained (Table 4). The total number of objects becomes 6.

The illustrations of this selected information table are given as $Y_{cache} = \{e_4, e_5, e_6, e_{17}\}$ and $Y_{notcache} = \emptyset$, and the domain is $E = \{e_4, e_5, e_6, e_{17}\}$. We locate the indiscernibility classes based on SD and RT as $\emptyset$. The lower approximation is $\underline{Y} = \cup_{\Omega_i \subseteq Y} \Omega_i = \emptyset$ and the upper approximation is $\overline{Y} = \cup_{\Omega_i \cap Y \neq \emptyset} \Omega_i = \{e_4, e_5, e_6, e_{17}\}$. The discriminant index of SD and RT is $\alpha_{C2,C3}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (6 - 0)/6 = 0$.

The indiscernibility classes based on TS and RT is $\{e_4, e_5, e_6, e_{17}\}$. The lower approximation is $\underline{Y} = \cup_{\Omega_i \subseteq Y} \Omega_i = \{e_4, e_5, e_6, e_{17}\}$ and the upper approximation is $\overline{Y} = \cup_{\Omega_i \cap Y \neq \emptyset} \Omega_i = \{e_4, e_5, e_6, e_{17}\}$. The discriminant index of TS and RT is $\alpha_{C1,C3}(Y) = 1 - |\overline{Y} - \underline{Y}|/|E| = 1 - (6 - 6)/6 = 1$.

By comparing the discriminant indices, we discover that $\alpha_{C1,C3}(Y) = 1$ best determines the membership in Y. Thus, we attain the sample of second rule:

R$_2$ **:** $\{$***Timestamp* = 790358527*, Objectretrievaltime* = *0***$\}\Rightarrow\{$***Cache* = *1***$\}$

Two rules have been found. If new domain is uncovered and new rules are computed using the same method as previous, then the irrelevant elements can be removed as $(E - \overline{Y}) \cup (\underline{Y}) = \emptyset \cup \{e_4, e_5, e_6, e_{17}\}$.

By referring to Table 3, we can see that the first set is empty and the second set has been handled by rule 2. Hence, the new set of elements becomes $(E - [(E - \overline{Y}) \cup (\underline{Y})] = \{e_4, e_5, e_6, e_{17}\}$.

Based on this assumption, we obtain supplementary collapsed information table in which SD and RT are omitted due to superfluous attributes (see Table 5).

The rules are fruitfully induced. A question that rises is how much we can believe in these rules. Therefore, we need to evaluate the strength of the rules as follows [61, 54]:

$$\frac{\#\text{ of positive objects covered by the rule}}{\#\text{ of objects covered by the rule (including both positive and negative)}}$$

Based on this equation, the first rule has strength of 6/20. It shows that 30% Classes of e_4, e_5, e_6, ande_{17} (Table 3) are positive examples covered by the rule. Class e_1 is a negative example covered by the first rule. The second rule has the strength of 6/6, that is, 100%. In applying the first rule to this object, there is a 30% chance that the reason for cache the object is exclusively the cache of RT. However, there is a higher probability that the reason for cache is due to extra timing of TS and RT, due to 100% strength of the second rule. Algorithm 1 illustrates the algorithm of rules induction using RS [57].

Algorithm 1. Rough set algorithm [57]

```
1: for each decision class do
2:     Initialise universe of objects
3:     Select decision class
4:     Find class relation
5:     repeat
6:         for each attribute do
7:             Select attribute
8:             Find equivalence relation
9:             Find lower subset
10:            Find upper subset
11:            Calculate discriminant index
12:        end for
13:        Select attribute with highest discriminant index
14:        Generate rules
15:        Reduce universe of objects
16:        Reduce class relation
17:    until no objects with selected decision class
18: end for
```

This part presents substantial RS analysis based on Inductive Learning methods to optimize Web caching performance to probe significant attributes and generate the decision rules. RS granularity in Web caching allows decision rules to be induced. These rules are important in optimizing user storage by executing caching strategy in specifying the most relevant condition attributes. This approach provides guidance to the administrator in Web caching regarding to selection of the best parameters to be cached. Based on this analysis, the administrator may reorganize the parameter of log data set in proxy caching accordingly.

6 Experimental Results

In this part describes experimental results of dataset for HTTP requests and user behavior of a set of Mosaic clients running in the Boston University (BU), Computer Science Department [47].

6.1 BU Log Dataset

In this experiment, BU Web Trace collected by Oceans Research Group at Boston University is employed. BU traces records consist of 9,633 files with a population of 762 different users, and recording 1,143,839 requests for data transfer. The data for January 1995 comprises of 11 to 220 users with 33,804 records. However, after data cleaning, only 10,727 dataset is left.

Moreover, in this research RS is exploited to reduce the rules of a log file and simultaneously to enhance the prediction performance of user behavior. RS is beneficial in probing the most significant attributes with crucial decision rules to facilitate intelligent caching and prefetching to safeguard limited bandwidth and minimize the processing cost.

The dataset is split in two; 70% (7,187 objects) for training and 30% (3,540 objects) for testing. To simplify data representation, a Naïve Discretization Algorithm (NA) is exploited and Genetic Algorithm (GA) is chosen to generate the object rules. Next, Standard Voting Classifier (SVC) is selected to classify the log file dataset. The derived rules from the training are used to test the effectiveness of the unseen data. In addition, 3-Fold Cross Validation is implemented for validation of our experiment. First fold (K1) the testing data from 1 to 3540, second fold (K2) from 3541 to 7081 and third fold (K3) from 7082 to 10622. Data are stored in decision table. Columns represent *attributes*, rows represent *objects* whereas every cell contains *attribute value* for corresponding objects and attributes. A set of attributes are *URL, Machinename, Timestamp, Useridno, Sizedocument, Objectretrievaltime*, and *Cache* as a decision.

6.2 Data Discretization and Reduction

Training data is discretized using NA. This discretization technique is implemented a very straightforward and simple heuristic that may result in very many cuts, probably far more than are desired. In the worst case, each observed value is assigned its own interval. GA is used for reduct generation [63] as it provides more exhaustive search of the search space. Reducts generation have two options [64]; full object reduction and object related reduction. Full object reduction produces set of minimal attributes subset that defines functional dependencies, while reduct with object related produce a set of decision rules or general pattern through minimal attributes subset that discern on a per object basis. The reduct with object related is preferred due to its capability in generating reduct based on discernibility function of each object.

Table 6 illustrates the comparison results of generation of a log file dataset in different K-fold (K1, K2 and K3). The highest testing accuracy is 98.46% achieved through NA discretization method and GA with full reduct method. Number of reducts for K1, K2 and K3 are equivalent. Object related reduct, 22 and full reduct, 6. In our observation, the highest number of rules are GA with full reduct, 63311 for K1, K2 and K3 and the highest testing accuracy is GA with full reduct for K1, 98.46%.

Table 6. Comparison reduct for K1, K2 and K3

Discretize Method	Reduct Method	K-fold	No.of Reduct	No.of Rules	Testing Accuracy (%)
NA	GA (object related)	K1	22	26758	96.8644
		K2	22	26496	96.8644
		K3	22	26496	96.8079
	GA (full object)	K1	6	63311	98.4618
		K2	6	63311	5.76271
		K3	6	63311	5.79096

6.3 Rule Derivation

A unique feature of the RS method is its generation of rules that played an important role in predicting the output. ROSETTA tool has listed the rules and provides some statistics for the rules which are support, accuracy, coverage, stability and length. Below is the definition of the rule statistics [64]:

- The rule LHS support is defined as the number of records in the training data that fully exhibit property described by the IF condition.
- The rule RHS support is defined as the number of records in the training data that fully exhibit the property described by the THEN condition.
- The rule RHS accuracy is defined as the number of RHS support divided by the number of LHS support.
- The rule LHS coverage is the fraction of the records that satisfied the IF conditions of the rule. It is obtained by dividing the support of the rule by the total number of records in the training sample.
- The rule RHS coverage is the fraction of the training records that satisfied the THEN conditions. It is obtained by dividing the support of the rule by the number of records in the training that satisfied the THEN condition.

The rule length is defined as the number of conditional elements in the IF part. Table 7 shows the sample of most significant rules. These rules are sorted according to their support value. The highest support value is resulted as the most significant rules. From the Table 7, the generated rule of {Sizedocument(0) $\Rightarrow$ Cache(1)} is considered the most significant rules with the outcome of not cache (output=0) and with cache (output=1). This is supported by 3806 for LHS support and RHS support value. Subsequently, the impact of rules length on testing accuracy are evaluated based on rules set from Table 7. Consequently, the same rules are divided into two groups; $1 \leq$ rules of length ≤ 2. It seems that the rules with length ≥ 1 contribute better classification compared to the rules with length ≤ 2.

Table 7. Sample for sorted of highest rule support values from data decision table for K1, K2 and K3

Rule	LHS Support	RHS Support	LHS Length	RHS Length
K1				
Sizedocument(0) ⇒ Cache(1)	3806	3806	1	1
Objectretrievaltime(0.000000) ⇒ Cache(1)	3805	3805	1	1
Sizedocument(2009) ⇒ Cache(0)	233	233	1	1
Sizedocument(717) ⇒ Cache(0)	128	128	1	1
K2				
URL(http://cs-www.bu.edu/lib/pics/bu-logo.gif) AND Sizedocument(0) ⇒ Cache(1)	1009	1009	2	1
URL(http://cs-www.bu.edu/lib/pics/bu-logo.gif) AND Objectretrievaltime(0.00000) ⇒ Cache(1)	1009	1009	2	1
Machinename(beaker) AND Sizedocument(0) ⇒ Cache(1)	308	308	2	1
Machinename(beaker) AND Objectretrievaltime(0.00000) ⇒ Cache(1)	308	308	2	1
K3				
URL(http://cs-www.bu.edu/lib/pics/bu-logo.gif) AND Objectretrievaltime(0.00000) ⇒ Cache(1)	989	989	2	1
URL(http://cs-www.bu.edu/lib/pics/bu-logo.gif) AND Sizedocument(0) ⇒ Cache(1)	989	989	2	1
Machinename(beaker) AND Sizedocument(0) ⇒ Cache(1)	306	306	2	1
Machinename(beaker) AND Objectretrievaltime(0.00000) ⇒ Cache(1)	306	306	2	1

6.4 Classification

From the analysis, it shows that the classification is better. Furthermore, the core attributes and the significant rules can improve the accuracy of classification. Table 8 shows the result of classification performance of K1, K2 and K3 for the original table and the new decision table of log file dataset. Hence, Figure 2 depicts an overall accuracy for log file, 36.67% for all rules in original decision table and 96.85% for selected rules in new decision table. This result shows a different of overall accuracy up to 60.18% between the original decision table and new decision table.

Table 8. Classification performance of K1, K2 and K3 for both original decision table and new decision table of log file dataset

Decision Table	Rule Set	K-fold	Accuracy (%)	Overall Accuracy (%)
New decision table	Selected rules	K1	96.8644	96.85
		K2	96.8644	
		K3	96.8079	
Orig. decision table	All rules	K1	98.4618	36.67
		K2	5.76271	
		K3	5.79096	

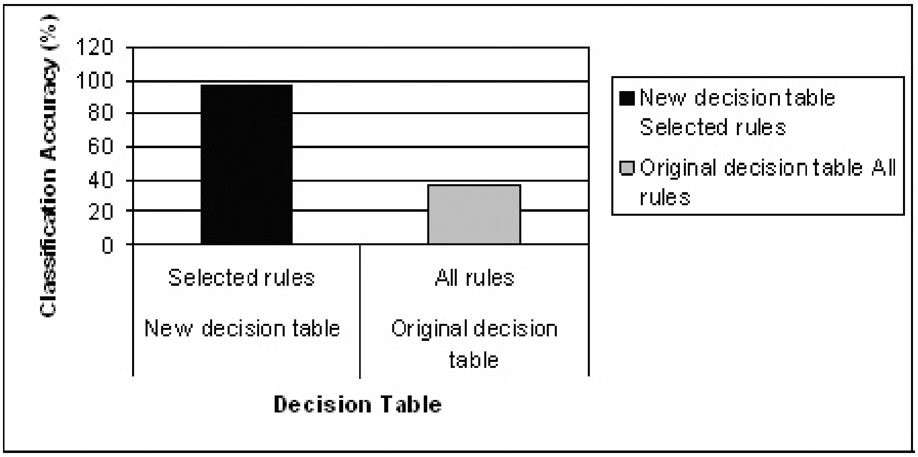

Fig. 2. Overall classification accuracy for both original decision table and new decision table of log file dataset

7 Conclusions

This chapter illustrated the usage of rough set theory for performance enhancement of Web caching. The RClass System framework [46] is used as a knowledge representation scheme for uncertainty in data for optimizing the performance of proxy caching that use to store the knowledge discovery of user behaviors in log format.

Furthermore, substantial RS analysis based on Inductive Learning methods is presented to optimize Web caching performance to probe significant attributes and generate the decision rules. RS granularity in Web caching allows decision rules to be induced. These rules are important in optimizing users' storage by executing caching strategy in specifying the most relevant condition attributes. This approach provides guidance to the administrator in Web caching regarding to selection of the best parameters to be cached. Based on this analysis, the administrator may reorganize the parameter of log data set in proxy caching accordingly.

Moreover, an empirical study has been conducted for searching optimal classification. A RS framework for log dataset is illustrated mutually with an analysis of reduced and derived rules, with entrenchment of their implicit properties for better classification outcomes.

In the future, more experiments on huge data will be conducted on hybridization of RS and evolutionary computation to deal with multiple knowledge of Web caching in reducing network latency.

Acknowledgements

This work is supported by MOSTI and RMC, Universiti Teknologi Malaysia, MALAYSIA. Authors would like to thank Soft Computing Research Group, Faculty of Computer Science and Information System, UTM for their continuous support and fondness in making this work an accomplishment.

References

1. Walter, J.D., Ahrvind, J.T.: The economic value of rapid response time. Technical Report GE20-0752-0, IBM, White Plains, NY (November 1982)
2. James, T.B.: A theory of productivity in the creative process. IEEE Computer Graphics and Applications 6(5), 25–34 (1986)
3. Chris, R.: Designing for delay in interactive information retrieval. Interacting with Computers 10, 87–104 (1998)
4. Judith, R., Alessandro, B., Jenny, P.: A psychological investigation of long retrieval times on the World Wide Web. Interacting with Computers 10, 77–86 (1998)
5. Nina, B., Anna, B., Allan, K. (2000) Integrating user perceived quality into Web server design. In: Proceedings of the Ninth International World Wide Web Conference, Amsterdam (May 2000)
6. Zona, The economic impacts of unacceptable Web site download speeds. White paper, Zona Research (1999), `http://www.zonaresearch.com/deliverables/whitepapers/wp17/index.htm`

7. Lu, J., Ruan, D., Zhang, G.: E-Service Intelligence Methodologies. In: Technologies and Applications. Springer, Heidelberg (2006)
8. Davison, B.D.: The Design and Evaluation of Web Prefetching and Caching Techniques. Doctor of Philosophy thesis, Graduate School of New Brunswick Rutgers, The State University of New Jersey, United State (2002)
9. Nagaraj, S.V.: Web Caching and Its Applications. Kluwer Academic Publishers, Dordrecht (2004)
10. Krishnamurthy, B., Rexford, J.: Web Protocols and Practice: HTTP 1.1, Networking Protocols. Caching and Traffic Measurement. Addison Wesley, Reading (2001)
11. Acharjee, U.: Personalized and Intelligence Web Caching and Prefetching. Master thesis, Faculty of Graduate and Postdoctoral Studies, University of Ottawa, Canada (2006)
12. Garg, A.: Reduction of Latency in the Web Using Prefetching and Caching. Doctor of Philosophy thesis, University of California, Los Angeles, United State (2003)
13. Kroeger, T.M., Long, D.D.E., Mogul, J.C.: Exploring The Bounds of Web Latency Reduction from Caching and Prefetching. In: Proceedings of the USENIX Symposium on Internet Technology and Systems, pp. 13–22 (1997)
14. Davison, B.D.: A Web Caching Primer. IEEE Internet Computing, pp. 38–45 (2001), `http://computer.org/internet`
15. Wong, K.Y., Yeung, K.H.: Site-Based Approach in Web Caching Design. IEEE Internet Comp. 5(5), 28–34 (2001)
16. Wong, K.Y.: Web Cache Replacement Policies: A Pragmatic Approach. IEEE Network (January/Feburary 2006)
17. Mohamed, F.: Intelligent Web Caching Architecture. Master thesis, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Malaysia (2007)
18. Mohamed, F., Shamsuddin, S.M.: Smart Web Caching with Structured Neural Networks. In: Proc. Of The 2nd National Conf. on Computer Graphics and Multimedia, Selangor, pp. 348–353 (2004)
19. Mohamed, F., Shamsuddin, S.M.: Smart Web Cache Prefetching with MLP Network. In: Proc. Of The 1st IMT-GT Regional Conf. On Mathematics, Statistics and their Applications, pp. 555–567 (2005)
20. Curran, K., Duffy, C.: Understanding and Reducing Web Delays. Int. J. Network Mgmt. 15, 89–102 (2005)
21. Web Caching, Caching Tutorial for Web Authors (2008), `http://www.web-caching.com/mnot_tutorial/intro.html`
22. Saiedian, M., Naeem, M.: Understanding and Reducing Web Delays. IEEE Computer Journal 34(12) (December 2001)
23. Foedero.com, Dynamic Caching (2006), `http://www.foedero.com/dynamicCaching.html`
24. Fan, L., Jacobson, Q., Cao, P., Lin, W.: Web prefetching between low-bandwidth clients and proxies: potential and performance. In: Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modelling of Computer Systems, Atlanta, Georgia, USA, pp. 178–187 (1999)
25. Yang, Q., Zhang, Z.: Model Based Predictive Prefetching. IEEE, 1529-4188/01: 291–295 (2001)
26. Yang, W., Zhang, H.H.: Integrating Web Prefetching and Caching Using Prediction Models. In: Proceedings of the 10th international conference on World Wide Web (2001)
27. Jiang, Z., Kleinrock, L.: Web Prefetching in a Mobile Environment. IEEE Personal Communications, 1070-9916/98: 25–34 (1998a)

28. Jiang, Z., Kleinrock, L.: An Adaptive Network Prefetch Scheme. IEEE Journal on Selected Areas in Communications 16(3), 1–11 (1998b)
29. Santhanakrishnan, G., Amer, A., Chrysanthis, P.K.: Towards Universal Mobile Caching. In: Proceedings of MobiDE 2005, Baltimore, Maryland, USA, pp. 73–80 (2005)
30. Ari, I., Amer, A., Gramacy, R., Miller, E.L., Brandt, S., Long, D.D.E.: Adaptive Caching using Multiple Experts. In: Proc. of the Workshop on Distributed Data and Structures (2002)
31. Teng, W.-G., Chang, C.-Y., Chen, M.-S.: Integrating Web Caching and Web Prefetching in Client-Side Proxies. IEEE Transaction on Parallel and Distributed Systems 16(5) (May 2005)
32. Kobayashi, H., Yu, S.-Z.: Performance Models of Web Caching and Prefetching for Wireless Internet Access. In: International Conference on Performance Evaluation: Theory, Techniques and Applications (PerETTA 2000) (2000)
33. Markatos, E.P., Chironaki, C.E.: A Top 10 Approach for Prefetching the Web. In: Proc. of INET 1998:Internet Global Summit (1998)
34. Duchamp, D.: Prefetching Hyperlinks. In: Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems(USITS 1999), Boulder, Colorado, USA (1999)
35. Deshpande, M., Karypis, G.: Selective Markov Models for Predicting Web-Page Accesses. In: Proceedings SIAM International Conference on Data Mining (2001)
36. Song, H., Cao, G.: Cache-Miss-Initiated Prefetch in Mobile Environments. Computer Communications 28(7) (2005)
37. Yin, L., Cao, G.: Adaptive Power-Aware Prefetch in Mobile Networks. IEEE Transactions on Wireless Communication 3(5) (September 2004)
38. Wu, S., Chang, C., Ho, S., Chao, H.: Rule-based intelligent adaptation in mobile information systems. Expert Syst. Appl. 34(2), 1078–1092 (2008)
39. Komninos, A., Dunlop, M.D.: A calendar based Internet content pre-caching agent for small computing devices. Pers Ubiquit Comput. (2007), DOI 10.1007/s00779-007-0153-4
40. Cao, P., Zhang, J., Beach, K.: Active Cache:Caching Dynamic Contents on The Web. Distributed Systems Engineering 6(1), 43–50 (1999)
41. Shi, Y., Watson, E., Chen, Y.-S.: Model-Driven Simulation of World-Wide-Web Cache Policies. In: Proceedings of the 1997 Winter Simulation Conference, pp. 1045–1052 (1997)
42. Abrams, M.: WWW:Beyond the Basics (1997), `http://ei.cs.vt.edu/~wwwbtb/fall.96/bppk/chap25/index.html`
43. Abrams, M., Standridge, C.R., Abdulla, G., Williams, S., Fox, E.A.: Caching proxies: Limitations and Potentials. In: Proceedings of the 4th International WWW Conference, Boston, MA (December 1995), `http://www.w3.org/pub/Conferences/WWW4/Papers/155/`
44. Zadeh, L.: Fuzzy sets. Information and Control 8, 338–353 (1965)
45. Pawlak, Z.: Rough Sets - Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
46. Triantaphyllou, E., Felici, G.: Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques. Massive Computing Series, pp. 359–394. Springer, Heidelberg (2006)
47. BU Web Trace, `http://ita.ee.lbl.gov/html/contrib/BU-Web-Client.html`
48. Wakaki, T., Itakura, H., Tamura, M., Motoda, H., Washio, T.: A Study on Rough Set-Aided Feature Selection for Automatic Web-page Classification. Web Intelligence and Agent Systems: An International Journal 4, 431–441 (2006)

49. Liang, A.H., Maguire, B., Johnson, J.: Rough Set WebCT Learning, pp. 425–436. Springer, Heidelberg (2000)
50. Ngo, C.L., Nguyen, H.S.: A Tolerence Rough Set Approach to Clustering Web Search Results, pp. 515–517. Springer, Heidelberg (2004)
51. Chimphlee, S., Salim, N., Ngadiman, M.S., Chimphlee, W., Srinoy, S.: Rough Sets Clustering and Markov model for Web Access Prediction. In: Proceedings of the Postgraduate Annual Research Seminar, Malaysia, pp. 470–475 (2006)
52. Khasawneh, N., Chan, C.-C.: Web Usage Mining Using Rough Sets. In: Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS 2005), pp. 580–585 (2005)
53. Pawlak, Z.: Rough Sets, pp. 3–8. Kluwer Academic Publishers, Dordrecht (1997)
54. Johnson, J., Liu, M.: Rough Sets for Informative Question Answering. In: Proceedings of the International Conference on Computing and Information (ICCI 1998), pp. 53–60 (1996)
55. Liang, A.H., Maguire, B., Johnson, J.: Rough set based webCT learning. In: Lu, H., Zhou, A. (eds.) WAIM 2000. LNCS, vol. 1846, pp. 425–436. Springer, Heidelberg (2000)
56. Johnson, J.A., Johnson, G.M.: Student Characteristics and Computer Programming Competency: A Correlational Analysis. Journal of Studies in Technical Careers 14, 23–92 (1992)
57. Tsaptsinos, D., Bell, M.G.: Medical Knowledge Mining Using Rough Set Theory, `http://citeseer.ist.psu.edu/86438.html`
58. Shan, N., Ziarko, W., Hamilton, H.J., Cercone, N.: Using rough sets as tools for knowledge discovery. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD 1995), pp. 263–268. AAAI Press, Menlo Park (1995)
59. Sulaiman, S., Shamsuddin, S.M., Abraham, A.: An Implementation of Rough Set in Optimizing Mobile Web Caching Performance. In: Tenth International Conference on Computer Modeling and Simulation, UKSiM/EUROSiM 2008, pp. 655–660. IEEE Computer Society Press, Los Alamitos (2008)
60. Sulaiman, S., Shamsuddin, S.M., Forkan, F., Abraham, A.: Intelligent Web Caching Using Neurocomputing and Particle Swarm Optimization Algorithm. In: Second Asia International Conference on Modeling and Simulation, AMS 2008, pp. 642–647. IEEE Computer Society Press, Los Alamitos (2008)
61. Pawlak, Z., Grzymala-Busse, J., Slowinski, R., Ziarko, W.: Rough sets. Communications of the ACM 38(11), 89–95 (1995)
62. Johnson, J., Liu, M.: Rough Sets for Informative Question Answering. In: Proceedings of the International Conference on Computing and Information (ICCI 1998), pp. 53–60 (1996)
63. Wróblewski, J.: Finding minimal reducts using genetic algorithms. In: Proceedings of Second International Joint Conference on Information Science, pp. 186–189 (1995)
64. Sulaiman, N.S.: Generation of Rough Set (RS) Significant Reducts and Rules for Cardiac Dataset Classification. Master thesis, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Malaysia (2007)