

Data Warehousing for Rough Web Caching and Pre-fetching

Sarina Sulaiman¹, Siti Mariyam Shamsuddin¹, Ajith Abraham²

¹Soft Computing Research Group (SCRG)

K-Economy Research Alliance (RAKE)

Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

²Machine Intelligence Research Labs (MIR Labs), Washington 98092, USA.

<http://www.mirlabs.org>

sarina@utm.my, mariyam@utm.my, ajith.abraham@ieee.org

Abstract

It is no secret that the Internet world has now evolved rapidly. From text-based to multimedia has congested our network traffic. This situation causes much lagging in the current workplace or geographic area. One of the infrastructures that support and convey data is a proxy server. Proxy server needs efficiently to use historical data to give users best links and also to reduce the traffic congestions. Furthermore, historical data are stored in a data warehouse for further processing. From the data warehouse, the best hit ratio can be obtained and, which increases the probability to keep the cache object for proxy server to serve the client faster. In order to get the hit ratio, rough set algorithms are applied to mark the lower and upper approximation and to get the best results.

1. Introduction

Caching and pre-fetching is middle-aged technology widely used in many areas such as Database Systems and Operating Systems. Presently, the World Wide Web becomes another attractive area in applying caching and pre-fetching.

The hit rate and byte hit rate are two extensively applied performance metrics in Web caching [1, 2]. Hit rate is the ratio of the number of requests that reach the proxy cache and the total number of requests. Byte hit rate is the ratio of the number of bytes that reach the proxy cache and the total number of bytes requested.

As Yang and Zhang has indicated: "Fractional network traffic is also defined to measure the increased network load [1].

Definition *Fractional latency* is the ratio between the observed latency with and without a caching or pre-fetching system.

Definition *Fractional network traffic* is the ratio between the number of bytes that are transmitted from Web servers to the proxy and the total number of bytes requested. Clearly, the lower the fractional latency and network traffic, the better the performance; for example, the fractional latency of 30% achieved by a caching system means the caching system saves 70% of the latency".

Web caching and Web pre-fetching are two essential techniques used to reduce the visible response time identified by users. In other words, by incorporating Web caching and Web pre-fetching, these two techniques can complement each other since the Web caching technique exploits the temporal locality, while Web pre-fetching technique exploits the spatial locality of Web objects [3]. Nevertheless, the integration of these two techniques might cause significant performance degradation to each other without cautious design.

Based on the actual fact, this paper proposes an implementation of a data warehouse for rough Web caching and pre-fetching, which not only considers the caching effect in the Web environment, but also evaluates the pre-fetching rules. Explicitly, a normalized profit function is formulated to evaluate the profit from caching an object either no-cache or cache object according to some pre-fetching rule. Teng et al. [3] used the normalized profit function devised an innovative Web cache replacement algorithm, so-called as Algorithm IWCP (standing for the Integration of Web Caching and Pre-fetching). They evaluate the performance of Algorithm IWCP under several circumstances by using an event-driven simulation.

The research will focus on Web proxy servers [3,13,14]. A proxy server is usually located at the edge of a LAN, intercepting HTTP requests and responses between clients and Web servers. Figure 1 depicts the

implementation of the proxy server between clients and servers.

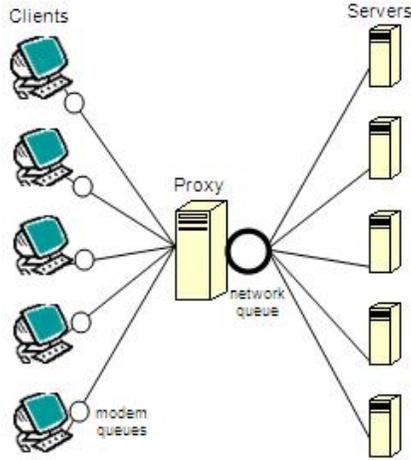


Figure 1. Proxy server implementation.

If it found that the requested object is already stored in its cache, returns the object to the user. Otherwise, it goes to the original server on behalf of the user, grabs the object, stores it in its cache, and returns the object to the user. An advantage of Web proxy caching and pre-fetching is that all clients within the Local Area Network (LAN) can share objects stored in the cache.

In order to calculate the benefits of pre-fetching, the common currency that will be used to measure the benefit must be selected. Since the ultimate goal of this study is to mask document latency, latency is selected as the currency and to attempt to minimize it.

According to Foygel and Strelow [4], “To simplify the analysis, it is assumed that the only latency associated with a document request is the network delay. Specifically, a linear network model was chosen, where the latency for a document of size S is

$$L(S) = N_C + N_B * S$$

(N_C is the constant network overhead and N_B is the constant describing the effective bandwidth of the network).

Moreover, it is assumed that there is no latency associated with locally stored documents. Hence, all the latencies are masked if the document can be found in the cache (or pre-fetch buffer)”.

Section 2 discusses the definitions related to this study. The pre-fetching and its correlation is described in Section 3. The methodology of this research is illustrated and explained in Section 4. Section 5 and 6 discusses experimental setup and results from this research. Finally, we conclude the work and outline some possible future work in Section 7.

2. Definitions

This section contains an explanation of the basic framework of rough set theory proposed originally, along with some of the key definitions.

A rough set first described by Zdzislaw Pawlak [5,6,7] a formal approximation of a crisp set (i.e., conventional set) in terms of a pair of sets, which give the lower and upper approximation of the original set. However, in other variations, the approximating sets may be fuzzy sets.

2.1. Information System Framework

Let $I = (\mathbb{U}, \mathbb{A})$ be an information system (attribute-value system), where \mathbb{U} is a non-empty set of finite objects (the universe) and \mathbb{A} is a non-empty, finite set of attributes such that $a : \mathbb{U} \rightarrow V_a$ for every $a \in \mathbb{A}$. V_a is the set of values that attribute a may take. In words: the information table assigns a value in V_a to each attribute a of each object in the universe \mathbb{U} .

With any $P \subseteq \mathbb{A}$ there is an associated equivalence relation $IND(P)$:

$$IND(P) = \{(x, y) \in \mathbb{U}^2 \mid \forall a \in P, a(x) = a(y)\}$$

The partition of \mathbb{U} generated by $IND(P)$ is denoted $\mathbb{U}/IND(P)$ (or \mathbb{U}/P) and can be calculated as follows:

$$\mathbb{U}/IND(P) = \otimes \{\mathbb{U}/IND(\{a\}) \mid a \in P\}, (x, y) \in IND(P)$$

Where

$$A \otimes B = \{X \cap Y \mid \forall X \in A, \forall Y \in B, X \cap Y \neq \emptyset\}$$

If $(x, y) \in IND(P)$, then x and y are indiscernible by attributes from P . In words: for any selected subset of attribute P , there will be sets of objects that are indiscernible based on those attributes. These indistinguishable sets of objects, therefore, define an equivalence or indiscernibility relation, referred to as the P -indiscernibility relation.

Present a prediction model based on statistical correlation between Web objects. An approach to integrate a pre-fetching and caching algorithm is proposed by incorporating pre-fetching into a Web caching algorithm. Particularly, this study tries to answer these questions:

1. How do we incorporate pre-fetching into a Web caching system?
2. How effective is the incorporated pre-fetching and caching algorithm?

Greedy Dual-Size (GDS) algorithm [8] computes its key value $K(p)$ as follows:

$$K(p) = L + C(p) / S(p)$$

Where

$C(p)$ is the cost to bring the object p into the cache;

$S(p)$ is the object size;

L is an inflation factor that starts at and is updated to the key value of the last replaced object. If an object is accessed again, its key value is updated using the new L value.

3. Pre-fetching and correlation

Pre-fetching is the operation of fetching information from remote Web servers even before it is requested. Objects such as images and hyperlink pages that are cited in a Web page (say a HTML page) may be fetched well before they are actually called for. It should be noted that a tradeoffs occurs. Web objects are pre-fetched assuming they will be requested soon. An accurate selection of such objects would lead to a reduction in the access latency, whereas inaccurate picks would only result in wasted bandwidth.

Correlation between Web objects can be obtained by studying the link structures of Websites. Furthermore, Web pages on related topics are often accessed together because of users' particular interests. The prediction algorithm in this study is based on the prediction model described by Padmanabhan and Mogul [9]. A remarkable difference is that our prediction model is time-based, which means the prediction window is a specific time period instead of a number of requests.

In order to present the links a correlation database is used to store information of pre-fetch information such as size, links, date access and protocols type. The database is divided into two tables, which are transaction and category as shown in Figure 2.

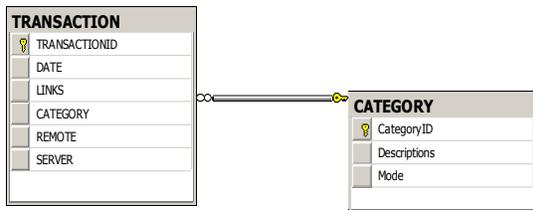


Figure 2. Database design.

The transaction table stores all links and category table is differentiating a type of transaction involved. This information can be analyzed further in a data warehouse. The data warehouse currently is implemented in Analysis Service SQL Server 2005.

4. Methodology

This study has implemented the rough set framework to optimize Web caching and pre-fetching (see Figure 3) [12]. The framework of the rough set tends to classify the data either to cache or not cache the

Web objects. It includes three main modules: a rough object consistency analyzer (ROCA), a rough object classifier (ROC) and a rough rule induction engine (RRIE) [12, 15, 16].

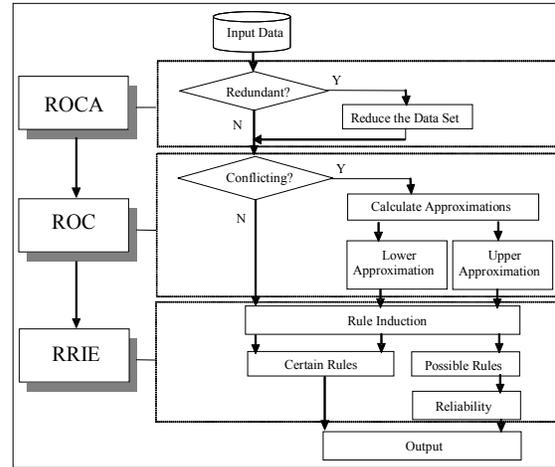


Figure 3. Rough set framework.

4.1. Rough Object Consistency Analyzer (ROCA)

The ROCA analyses the training data and performs two tasks; elimination of redundant data items, and identification of conflicting training data.

Cleaning Up

In order to make sure the data set is no crisp, any record set, which is empty will be removed to reduce the uncertainty automatically. For an instant, the LINKS columns may appear empty and RETURN TIME has null values. For this purpose, the rows are removed from the rough set approximate.

4.2. Rough Object Classifier (ROC)

The ROC has two approximator; the upper approximator and the lower approximator. The rough classifier is employed to treat inconsistent training data set (DS) in a duration (T). U is a non-empty finite set of an object and A is non-empty finite set of attributes as such that $a:U \rightarrow V_a$ is called the values set of $a.V_a$.

$$DS:T(U, A \cup \{d\}), d \in A$$

Indiscernibility

The non-empty subsets of the condition attribute are Hit Size, Links and Return Time, are represent by H, L and R respectively.

Approximation

The main goal of the rough set analysis is induction of approximations of concepts.

$$\begin{aligned} T &= (U,A) \\ B &\subseteq A \\ x &\subseteq U \end{aligned}$$

Now, to get the boundary of upper and lower boundary, we need to decide which characteristic to do a rough set on. For an instance, the required class decision for decision making involves the Hit Return for which Links. So, the rough set will use Lower Boundary

$$\{x | [x]_b \subseteq x\} = L.H$$

Upper Boundary

$$\{x | [x]_b \cap x\} = L.H \neq \emptyset$$

B-Boundary region of X

$$BN_b(X) = \overline{Bx} - Bx$$

Rough set bound is non-empty otherwise set crisp. The decision class, hit return is rough since the boundary region is not empty. For further explanations refer to [17].

4.3. Rough Rule Induction Engine (RRIE)

The RRIE module is implemented ID3-like learning algorithm, which is C4.5-based on the minimum-entropy principle. The concept of entropy is used to measure how informative an attribute is.

$$I_E(i) = - \sum_{j=1}^m f(i, j) \log_2 f(i, j).$$

The ID3 algorithm can be summarized as follows:

1. Take all unused attributes and count their entropy concerning test samples
2. Choose an attribute for which entropy is maximum
3. Make a node containing that attribute

5. Experimental setup

In this experiment, we use Boston University Web Trace [10] collected by Oceans Research Group at Boston University (BU) as our experiment data set. BU trace records are collected of 9,633 files, instead of a population of 762 different users, and recording 1,143,839 requests for data transfer. Figure 4 depicts the sample log data.

5.1. Method use reduct and core

The set of attributes is called a *reduct* of C , if $T' = (U, R, D)$ is independent and

$$POS_R(D) = POS_C(D).$$

The set of all the condition attributes indispensable in T is denoted by $CORE(C)$.

$$CORE(C) = \cap RED(C)$$

Where $RED(C)$ is the set of all *reduct* of C .

NO	NAME	TIME_STAMP	ID	SIZE	RET_TIME	LINKS
1	cs17	786146567	716362	1935	0.647182	http://cs-www.bu.edu/
2	cs17	786146568	763949	1804	0.742532	http://cs-www.bu.edu/lib/pics/bu-logo.gif
3	cs17	786146569	861378	716	0.302112	http://cs-www.bu.edu/lib/pics/bu-label.gif
4	cs17	786146580	438775	15745	0.919702	http://cs-www.bu.edu/faculty/bulletin/home.html
5	cs17	786146591	383338	0	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
6	cs17	786146591	386971	0	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
7	cs17	786146581	619336	21708	0.668233	http://cs-www.bu.edu/faculty/bulletin/pictures/bes
8	cs17	786146583	131665	14035	0.494767	http://cs-www.bu.edu/faculty/bulletin/pictures/che
9	cs17	786146587	264795	0	0	http://cs-www.bu.edu/80/
10	cs17	786146587	295778	0	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
11	cs17	786146587	297227	0	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
12	cs17	786146588	516869	2153	0.526822	http://cs-www.bu.edu/courses/home.html
13	cs17	786146589	150634	0	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
14	cs17	786146589	152272	0	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
15	cs17	786146590	448880	1805	0.416427	http://cs-www.bu.edu/courses/cs101a1/Home.html
16	cs17	786146590	355438	1803	0.312275	http://cs-www.bu.edu/courses/cs101a1/bu-logo.gif
17	cs17	786146593	207045	8107	0.326257	http://cs-www.bu.edu/courses/cs101a1/HW.html
18	cs17	786146825	815697	0	0	http://cs-www.bu.edu/courses/cs101a1/Home.html
19	cs17	786146825	846779	0	0	http://cs-www.bu.edu/courses/cs101a1/bu-logo.gif
20	cs17	786146827	154936	0	0	http://cs-www.bu.edu/courses/home.html
21	cs17	786146827	180493	0	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif

Figure 4. Sample data.

Scenario 1: Comparing Links Type and Return Time

Based on the SQL statement to retrieve the core, there are 3 types of protocols and these each records also needs to be categorized for which site links (see Figure 5). The indiscernibility classes defined by

$R = \{Links, Return Time.\}$ are

$X1 = \{u | links(u) = http\}$

```
select NAME,round(RET_TIME,2)AS ret_avg,LINKS
from links
where links like 'http%'
```

$X2 = \{u | links(u) = news\}$

```
select NAME,round(RET_TIME,2)AS ret_avg,LINKS
from links
where links like 'news%'
```

$X3 = \{u | links(u) = gopher\}$

```
select NAME,round(RET_TIME,2)AS ret_avg,LINKS
from links
where links like 'gopher%'
```

NAME	RET_TIME	LINKS
cs17	0.647182	http://cs-www.bu.edu/
cs17	0.742532	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	0.302112	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	0.919702	http://cs-www.bu.edu/faculty/bulletin/Home.html
cs17	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	0.668233	http://cs-www.bu.edu/faculty/bulletin/pictures/bes
cs17	0.494767	http://cs-www.bu.edu/faculty/bulletin/pictures/che
cs17	0	http://cs-www.bu.edu/80/
cs17	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	0.526822	http://cs-www.bu.edu/courses/Home.html
cs17	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	0.416427	http://cs-www.bu.edu/courses/cs101a1/Home.html
cs17	0.312275	http://cs-www.bu.edu/courses/cs101a1/bu-logo.gif
cs17	0.326257	http://cs-www.bu.edu/courses/cs101a1/HW.html
cs17	0	http://cs-www.bu.edu/courses/cs101a1/Home.html

Figure 5. Comparison 1.

Scenario 2: Comparing Links Size and Links

Based on the SQL statement to retrieve the core 3 types of protocols, each record also needs to be categorized in Size and Links (see Figure 6).

select NAME,SIZE,LINKS from links
 where links like 'http%' and SIZE > 2

NAME	SIZE	LINKS
cs17	1935	http://cs-www.bu.edu/
cs17	1804	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	716	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	15745	http://cs-www.bu.edu/faculty/bulletin/Home.html
cs17	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	21708	http://cs-www.bu.edu/faculty/bulletin/pictures/bes
cs17	14055	http://cs-www.bu.edu/faculty/bulletin/pictures/che
cs17	0	http://cs-www.bu.edu:80/
cs17	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	2153	http://cs-www.bu.edu/courses/Home.html
cs17	0	http://cs-www.bu.edu/lib/pics/bu-logo.gif
cs17	0	http://cs-www.bu.edu/lib/pics/bu-label.gif
cs17	1805	http://cs-www.bu.edu/courses/cs101a1/Home.h...

Figure 6. Comparison 2.

Each link may have been access many times over a certain period of time. By analyzing the different return time, a table is created. These records are divided into 2 halves; upper, and lower approximator.

6. Result

Currently, the program has been developed to obtain this result based on the calculation of rough set. These results were calculated from a set of data from BU log data from Nov 1994- May 1995 [10]. Cunha et al. [11] considered the number of sessions per day, and they also observed local and remote based on the frequencies of links for each Website as depicted in Figure 7 and the most popular sites based on a hit ratio for different protocol (see Figure 8) and each link (see Figure 9).

Hit Ratio Top 10		
	PROTOCOL	HIT
▶	http	7286
	ftp	179
	file	23
	gopher	236
	news	8

Figure 8. Hit ratio based on different protocol.

By using the data warehouse, the result obtained based on 3 core types of protocols, size hit frequency, count of each browser and link used by the diverse users as shown in Figure 10. Frequencies in the chart show the collectives and potential hit ratio that have been calculated from rough set. This information can

be used to develop further and enhance the capability proxy server to judge achievable Web links, which are commonly used. Links are normal and very subjective based on users' interaction to the Web browser. These interactions will be calculated and further refined under a lower and high boundary. These generated boundaries can be used to show the best links that matches the criteria search or information. These links are also subjected to the Transmission Control Protocols (TCP), which differentiates the best selections (refer to Figure 8).

7. Conclusion and future work

The results presented have managed to determine the possible Web pre-fetching according to hit ratio and hit links. Furthermore, it narrows down and increases the performances on overall links, protocol and size byte rather than just links. In order to incorporate pre-fetching into a Web caching system, proxy server should have the pre-fetch engine to narrow down the possible links at the local level. The effectiveness of this pre-fetching and caching algorithm depends and may vary on the usage and the type of search that is available. For example, Google has implemented the most frequent user search. By the time user finishes typing the word they are looking for, a list of all the hits is listed.

This algorithm can be well implemented in a social network Website as well such as Facebook, Twitter, MySpace and, etc. Users can use many applications in the social network Website, where sometime they may lose their way using it. By implementing this algorithm, social network Website users can benefit these criteria other than implementing it in a proxy server alone. The implementation of the mechanisms as seen can be widely used to refine and improve the access to many million Websites in the world as a proxy server.

8. Acknowledgment

The authors would like to thank Ministry of Science, Technology and Innovation Malaysia (MOSTI) and Reseach Management Centre (RMC), UTM for their kind financial support under the e-science fund. This paper also has benefited greatly from detailed feedback and constructive comments by the anonymous reviewers from SMC 2010. We are deeply grateful for Sheikh Nasir Kamarudin for providing helpful and kindness support to realize this research.

9. References

- [1] Q. Yang and Z. Zhang, "Model Based Predictive Prefetching," IEEE, 1529-4188/01, 2001, pp. 291-295.
- [2] Y. Zhu and Y. Hu, "Exploiting client caches to build large Web caches," The Journal of Supercomputing. 39(2), 2007, pp.149-175. Springer Netherlands.
- [3] W. Teng, C. Chang, and M. Chen, "Integrating Web Caching and Web Prefetching in Client-Side Proxies". IEEE Trans. Parallel Distrib. Syst. 16, 5 (May. 2005), pp.444-455. DOI= <http://dx.doi.org/10.1109/TPDS.2005.56>.
- [4] D. Foygel and D. Strelow, "Reducing Web latency with hierarchical cache based prefetching", Proceedings of 2000 International Workshop on Parallel Processing, Toronto, Ont., Canada, 2000, pp. 103-108.
- [5] J. Johnson, and M. Liu, "Rough Sets for Informative Question Answering," In Proceedings of the International Conference on Computing and Information (ICCI'98),Winnipeg, Canada, June 17-20 1996, pp. 53-60.
- [6] Z. Pawlak, "Rough Sets," International Journal of Computer and Information Sciences, vol.11, pp.341-356,1982.
- [7] Z. Pawlak, "Rough Sets: Theoretical Aspect of Reasoning about Data," Kluwer Academic Publishers, 1991.
- [8] P.Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," In Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97), Monterey, CA, December 1997, pp. 193-206.
- [9] V. N. Padmanabhan, and J. C. Mogul, "Using predictive prefetching to improve World Wide Web latency," SIGCOMM Comput. Commun. Rev. 26, 3 (Jul. 1996), pp.22-36. DOI= <http://doi.acm.org/10.1145/235160.235164>.
- [10] BU Web Trace, <http://ita.ee.lbl.gov/html/contrib/BUWeb-Client.html>.
- [11] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW Client-Based Traces," Technical Report. UMI Order Number: 1995-010, Boston University, 1995.
- [12] E. Triantaphyllou, and G. Felici, "Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques," Massive Computing Series, Springer,Heidelberg, Germany, 2006, pp.359-394.
- [13] R. Cáceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovich, "Web proxy caching: the devil is in the details," SIGMETRICS Perform. Eval. Rev. 26, 3 (Dec. 1998), 11-15. DOI= <http://doi.acm.org/10.1145/306225.306230>.
- [14] G. Banga, F. Douglis and M. Rabinovich, "Optimistic deltas for WWW latency reduction," In Proceedings of 1997 USENIX Technical Conference, 1997, pp.1-15.
- [15] S. Sulaiman, S.M. Shamsuddin and A. Abraham, "An Implementation of Rough Set in Optimizing Mobile Web Caching Performance," Tenth International Conference on Computer Modeling and Simulation, UKSiM/EUROSiM 2008, Cambridge, UK, IEEE Computer Society Press, USA, ISBN 0-7695-3114-8, 2008, pp. 655-660.
- [16] S. Sulaiman, S.M. Shamsuddin and A. Abraham, "Rough Web Caching," Book Chapter Rough Set Theory: A True Landmark in Data Analysis,2009, pp. 187-211, http://doi.acm.org/10.1007/978-3-540-89921-1_7.
- [17] S. Sulaiman, S.M. Shamsuddin, A. Abraham, and S. Sulaiman, "Rough Set Granularity in Mobile Web Pre-Caching," Eighth International Conference on Intelligent Systems Design and Applications - ISDA 2008, Taiwan, IEEE CS Press, USA, 2008, pp. 587-592.

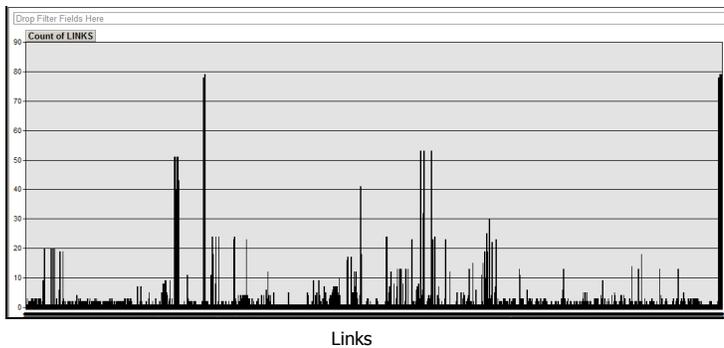


Figure 7. Frequencies of Links.

Hit Ratio TOP 10				
LINKS	HITRATIO	Descriptions	Mode	
http://sunsite.unc.edu/ouvrne/and/au/trauss/trau	1186.219971	HTTP	TCPIP	
http://multivac.ludd.luth.se/pub/sounds/songs/stra	879.1785600000...	HTTP	TCPIP	
http://lummi.stanford.edu/Media2/texts/mmdb26.ps	718.429138	HTTP	TCPIP	
http://multivac.ludd.luth.se/pub/sounds/songs/ub40	669.7313869999...	HTTP	TCPIP	
http://lummi.stanford.edu/Media2/texts/mmsj.ps	646.975159	HTTP	TCPIP	
http://multivac.ludd.luth.se/pub/sounds/songs/don_	641.6773835	HTTP	TCPIP	
http://multivac.ludd.luth.se/pub/sounds/songs/pear	577.618163	HTTP	TCPIP	
http://sunsite.unc.edu/ouvrne/and/au/O_Fortuna.au	482.868073	HTTP	TCPIP	
http://multivac.ludd.luth.se/pub/sounds/songs/emf/	480.9324253333...	HTTP	TCPIP	
http://www.spb.au/otr_spb/covr.gif	441.050049	HTTP	TCPIP	

Figure 9. Hit ratio for each Links.

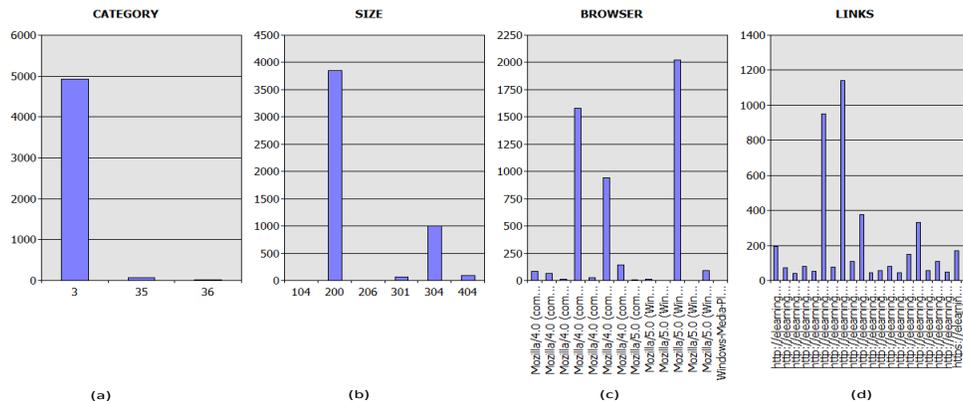


Figure 10. Graph representation for (a) Category, (b) Size hit Frequency, (c) Browser and (d) Links