**ANALYSIS OF WEB PREFETCHING TECHNIQUES**

COEN 283 Winter 2017

**Team Name: TAPOS**

**Team Members:**

Ankita Batni

Kuriakose Tintu

Prajakta Deosthali

**Tasks and Responsibilities**

|  |  |  |
| --- | --- | --- |
| Name | Tasks | Percentage |
| Ankita Batni | * Introduction and Data Prefetching * Formal Proposal * Report * Presentation | 33.3% |
| Kuriakose Mattathil Tintu | * Link and Predictive Prefetching * Formal Proposal * Report * Presentation | 33.3% |
| Prajakta Deosthali | * Semantic Prefetching and Conclusion * Formal Proposal * Report * Presentation | 33.3% |

**Preface**

As part of the curriculum and in order to gain practical knowledge in Operating System concepts, we are required to make a report on ‘Analysis of prefetching web proxy’.The basic objective behind doing this project report is to gain understanding of what is prefetching, why is it required and what are the various prefetching techniques. The project attempts to understand how various web prefetching techniques help in improving user experience over the web and also evaluate their effectiveness based on multiple factors. Apart from technical concepts, this project also helped us to learn about the importance of teamwork and the dynamics of working in a team.

**Acknowledgement**

We would like to express our gratitude to Santa Clara University for giving us this opportunity to work on this project. We would like to thank Professor Amr Elkady in Graduate Engineering Department of Santa Clara University for assigning this project.

**Table of Contents**

Abstract ...…………………………………….…………………………………....8

Chapter 1 - Introduction ..………………………..………………………………...9

1.1 What is Prefetching .……………….……………………………..……..9

1.2 Benefits of Prefetching .………………………………………………..10

1.3 Effective Prefetching.……….………..……...........................................11

1.4 Types of Prefetching.……….………..……............................................11

Chapter 2 - Link Prefetching.……………………………….………………….....13

2.1 Standard Link prefetching….....…….....…….....…………………........13

2.2 DNS prefetching………………….………….…….……...……............14

2.3 Prerendering……………………………..……...….…….....……….....15

2.4 Lazy load……………………………..……....…….....……………..…15

2.4.1 Benefits of Lazy Loading………...…………………................16

2.5 Sites using prefetching………..……………………..……...….………17

2.6 Evaluation of Link Prefetching……………………..……...….……....17

2.6.1 Client/Server……………………..……...….……………....…17

2.6.2 Effect on Bandwidth……………………..……...…………….18

2.6.3 Security……………………..……...………………………….18

Chapter 3 -Predictive Prefetching……………………………………………....…19

3.1 Association rules…………..………………..…….....…….....…….......19

3.2 Clustering.…….……...……...................................................................20

3.3 Markov Prediction ….…….....……........................................................20

3.4 Proxy side predictive prefetching engine (PPPE)….……...…………...21

3.5 Search engine side predictive prefetching engine (SPPE)….………….27

3.6 Evaluation of predictive prefetching engine….……...………………...31

3.6.1 Client/Server…………………………….…………………….31

3.6.2 Effect on bandwidth…………………………….……………..31

3.6.3 Accuracy…………………………….………………………...31

3.6.4 Timeliness…………………………….……………………….31

Chapter 4 - Data Prefetching..……………………………….…………………. ..32

4.1 Prediction method …………..………………..…….....…….....……....33

4.2 Data Prefetching in mobile applications.…….……...……......………..35

4.3 Evaluation ……………………………..……...….…….....……...........36

Chapter 5 -Semantic Prefetching………………………………………………….37

5.1 Lexical Analyzer……………………………………………………….37

5.2 Predictor Unit ………………………………………………………….38

5.3 Prefetch Unit …………………………………………………………..40

5.4 Storage Unit …………………………………………………………...40

5.5 Analysis of Semantic web prefetching………………………………...42

Chapter 6 -Design Considerations………………………………………………..45

Conclusion ……………………………………………………………………….47

References ………………………………………………………………………..48

**Table of Figures**

Figure 2.1: Standard Link Prefetching ....................................................................14

Figure 3.1: Navigational Graph ..............................................................................20

Figure 3.2: Framework for proxy side prefetching engine(PPPE) .........................23

Figure 3.3: Sample weblog before preprocessing ………………………………..24

Figure 3.4: Sample weblog after preprocessing…………………………………..25

Figure 3.5: Framework for search engine side prefetching engine(PPPE)..............28

Figure 3.6: Lower approximation set after application of RST ………………......30

Figure 4.1: Sequence of HTML documents requested .........................................33

Figure 5.1: Semantic web prefetching block diagram…………………………....37

Figure 5.2: Precision vs Data ..................................................................................39

Figure 5.3: Prefetch unit..........................................................................................40

Figure 5.4: Page session analysis ............................................................................43

Figure 5.5: User session time analysis ....................................................................45

**Abstract**

Web traffic has tremendously increased over the last 10 years. The number of websites in the world has reached the billion mark. With the rapid changes in technology and number of choices provided, user expectations are increasing as well. A click on your phone and your ride is on the way. Bad user experiences can make or break businesses. The time taken for web content to be delivered and loaded on the device after the request is made plays a major role in the end user experience. It has become as important as the way data is displayed on the users device, if not more. Prefetching is one of most experimented techniques for improving the speed at which data is delivered over the web. Over the years, several algorithms have been developed for web prefetching. Not all of these are alike and may not solve every use-case. As we enter into the Internet of Things era, new approaches might be needed. This project analyzes the existing approaches used for web prefetching and evaluates their effectiveness for different use-cases.

**Chapter 1**

**Introduction**

In recent years, the web has become the primary means for information dissemination. It is being used for commercial, entertainment, or educational purposes. Since the Internet capacity is not keeping pace, the net effect of this growth was a significant increase in the user perceived latency, that is, the time between when a client issues a request for a document and the time the response arrives. Potential sources of latency are the Web servers' heavy load, network congestion, low bandwidth, bandwidth underutilization, and propagation delay.

**1.1 What is Prefetching?**

​Prefetching is a performance optimization tactic in which content that might be accessed by the user is downloaded in advance. Unlike the static, text-based websites of the early Internet, today’s websites are driven by dynamic content and multimedia. A single web page can require multiple images and videos to be cached, resulting in long wait times and poor user experience.

Prefetching resolves this issue by predicting which links the user is likely to click, then downloading the content of those links. If the user decides to click on one of the links, then the page will be rendered instantly as the content has already been downloaded.

**1.2 Benefits of prefetching:**

* ​Prefetching ultimately results in a more fluid end user experience.
* Users experience faster load times for web content as pages are downloaded in the background. There’s no interruption to the user’s current browsing experience.
* Administrators see less spiked bandwidth usage since the content the user is likely to see has already been downloaded.
* Resources are delivered as needed, rather than all at once.
* Enterprises see higher visitor retainment as visitors no longer need to wait to access a web page.
* The browsing experience feels like the website is being hosted locally.

**1.3 Effective Prefetching:**

Prefetching only benefits the user and the provider if there is spare memory bandwidth to begin with, otherwise prefetches could make things worse by hogging up on the user’s bandwidth . The prefetches should be accurate, it makes sense only to prefetch data which will soon be used. The prefetches should be timely ie., prefetch the right data, but not too early as it would become stale soon and won’t be used, on the other hand if prefetches are too late then it defies the whole purpose of prefetching. Prefetching hides the latency which outweighs their cost , but cost of many useless prefetches could be significant.

So, the goal should be to keep the prefetches accurate and timely, avoid unnecessary prefetches and utilize the bandwidth optimally.

**1.4 Types of Prefetching**

There are different types of prefetching techniques. The techniques are classified based on various factors. It could be based on the location where the prefetched data is stored. It could also be based on the type of data prefetched, whether it is image, text or some other content.

* The prefetching techniques can be classified based on the access patterns:
  + Short Term Prefetching - The future request of the user are predicted based on the recent access history. Following techniquest come in this category :
    - Predictive
    - Semantic
  + Long Term prefetching - In this type, global access patterns are analyzed to predict the future requests. The web pages or web objects having high access count and recent updates are considered more likely to be requested. Following techniques fall under this category:
    - Frequency based prefetching
    - Greedy dual size prefetching
* The prefetching techniques can also be classified based on the location of storage:
  + Client side prefetching - The prefetched pages are stored on client’s machine. E.g. semantic web prefetching
  + Server side prefetching - The prefetched web pages are stored on the proxy server, for easy retrieval and to reduce latency. E.g. Predictive web prefetching

This report covers following types of prefetching techniques

1. Link Prefetching
2. Data prefetching
3. Predictive prefetching
4. Semantic prefetching

**Chapter 2**

**Link Prefetching**

Link prefetching is a mechanism for web pages to declare hints, allowing web browsers to pre-load various external resources to speed up loading and rendering of the page. The resources can include [JavaScript](https://en.wikipedia.org/wiki/JavaScript), image, audio, video or [web font](https://en.wikipedia.org/wiki/Web_typography) files as well as [DNS](https://en.wikipedia.org/wiki/Domain_Name_System) names. A web page supplies a set of prefetching hints to the browser, and once the browser finishes loading the page, it begins to prefetch the specified documents and stores them in its cache.

When a user visits a website, the client and server communicate in the background to load the [content](https://en.onpage.org/wiki/Content) as quickly as possible. Once the site is fully loaded, the browser goes idle. It only communicates with the server when the user interacts through clicks or keyboard commands. Link prefetching is executed during this idle time.

Depending on the browsers, there are various mechanisms that allow prefetching of resources.

1. Standard link prefetching
2. DNS prefetching
3. Prerendering
4. Lazy-load (Images)

**2.1 Standard Link prefetching:**

In link prefetching, resources are marked up with the prefetch value and the rel attribute, i.e., "rel="prefetch".

For example <link rel="prefetch" href="http://www.example.com/">

Only items that are marked up using this attribute-value pair can be loaded. In most cases, these are links to other webpages, images, or CSS files. We should note that only the referenced resource is loaded and not the files that are associated with this resource. No scripts, CSS files or frames gets loaded, which are located on the pre-loaded webpage. This type of prefetching was used in 2003 only by Mozilla Firefox. Since then standard link prefetching is executed by most modern browsers.

Below fig 2.1 shows how the prefetching content of most likely clicked links are loaded at the background before the user clicks on it.

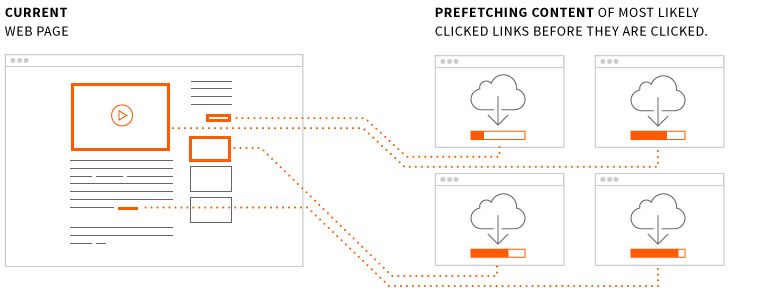


Fig 2.1: Standard Link Prefetching

**2.2 DNS prefetching**

DNS prefetching translates a DNS server into computer-readable characters (IP addresses) to use the idle time of the browser effectively. DNS resolution is therefore no longer required when loading the website. Prefetching of domain names is possible with the command "rel="dns-prefetch""

The syntax for these link tags is as follows:

<link rel="dns-prefetch" href="www.example.com">

Even though it can yield marginal gains in download performance for external assets, such as fonts, analytics, and third-party APIs, DNS Prefetching is clearly not a one-size-fits-all type of optimization. Depending on the nature of your site and users' navigation patterns, it can either be of great benefit or a drain on your ISP bandwidth allocations. So we should be aware of DNS Prefetching implications and decide for yourself how much of “ if any” it to utilize on your own site.

**2.3 Prerendering**

Prerendering is much different than prefetching. All resources that have been referenced by adding "prerender" to a rel attribute in a <link> element

For example, <link rel="prerender" href="prerender-this-page.html"> are loaded into the cache. This includes CSS, JavaScript and even the DOM.

Prerendering is similar to loading a website in a hidden browser tab. The complete resources are loaded. This is useful if the website owner is 100 % sure that the user will visit that website, otherwise the client will download all of the resources necessary to render the page even though they are not used at all.

**2.4 Lazy load**

when a user opens a webpage, the entire page’s contents are downloaded and rendered in a single go. While this allows the browser to cache the web page, there’s no guarantee that the user will actually view all of the downloaded content.For example, If an entire photo gallery is downloaded but the user leaves after only viewing the first image, then the result is wasted memory and bandwidth.Instead of bulk loading all of the content when the page is accessed, content can be loaded when the user accesses a part of the page that requires it. With lazy loading, pages are created with placeholder content which is only replaced with actual content when the user needs it.

When someone adds a resource to a web page (image, video, etc.), the resource references a small placeholder. As a user browses the web page, the actual resource is cached by the browser and replaces the placeholder when the resource becomes visible on the user’s screen. For instance, if the user loads a webpage and immediately leaves it, then nothing beyond the top portion of the web page is loaded.

**2.4.1 Benefits of Lazy Loading**

1. Lazy loading strikes a balance between optimizing content delivery and streamlining the end user’s experience.
2. Users connect to content faster, since only a portion of the website needs to be downloaded when a user first opens it.
3. Businesses see higher customer retention as content is continuously fed to the user, reducing the chances the user will leave the website.
4. Businesses see lower resource costs since content is only delivered when the user needs it, rather than all at once.

Provided an example where Internet explorer browser can load images in advance, which have been marked up with the lazy load tag. This is usually done within the img tag behind the source, which is specified with the src tag:

"<img src ="image.jpg"lazyload>"

Adding lazy loading can make a website load faster, save bandwidth, and provide a truly uninterrupted browsing session. [Almost half](http://www.gorocketfuel.com/the-rocket-blog/whats-the-average-bounce-rate-in-google-analytics/) of all users who visit a website view only a single page.

**2.5 Sites using prefetching**

Google is the most well-known [website](https://en.wikipedia.org/wiki/Website) that takes advantage of this feature so as to improve the user experience. For example, consider the search results returned for a particular query by the user, and if the first hit is considered very probable to be the desired hit, it is assigned as a prefetchable link.

[Google](https://en.wikipedia.org/wiki/Google_(search_engine)) implemented "Google Instant Pages" in August 2011 which introduced predictive search completion (preloading search results as the user types in order to improve the performance of the search engine)

[Bing](https://en.wikipedia.org/wiki/Bing_(search_engine)) has also employed the use of prefetching and preloading since the release of [Internet Explorer 11](https://en.wikipedia.org/wiki/Internet_Explorer_11) (October, 2013)

**2.6. Evaluation of Link Prefetching**

**2.6.1. Client/Server:** A server-driven approach to link prefetching is chosen because the servers (including intermediate proxy servers) can better predict what users are likely to visit next. Since pages often contain numerous hyperlinks that the user is unlikely to follow, it is generally difficult for the browser to determine the best subset of hyperlinks to prefetch.

For example, the server may want to instruct the browser to prefetch the most popular URLs on a given day. Because this technique works with HTTP headers, a proxy can very well generate such dynamically determined directives for popular links.

**2.6.2 Effect on Bandwidth:** Too many prefetching instructions can strain the server being accessed.If you are downloading something using Mozilla, link prefetching will be delayed until any background downloads complete. For example, if you load a bookmark group (which opens several tabs), any prefetch requests initiated by one of the bookmarked pages will not begin until all of the tabs finish loading. If you are using another application which uses the network, link prefetching in Mozilla may compete for bandwidth with the other applications. This is a problem that we hope to address in the future by leveraging operating system services to monitor network idle time.

**2.6.3 Security:** Security holes may arise because the prefetch instructions often refer to external providers. This means that other websites which were retrieved, may have been exploited for Distributed Denial of service(Ddos) attacks.

A DDoS attack is carried out in various ways, but there are generally three types:

* Overloading the server resources: In this case, the resources of the target system are overloaded that the server cannot answer any more queries.
* Overloading the bandwidth: This overloads the entire capacity of the network so that the target website is no longer accessible.
* By exploiting vulnerabilities in the server or website software, the attacker try to control the targeted server.

**Chapter 3**

**Predictive Web Prefetching**

Predictive web prefetching is a technique in which pages likely to be accessed by user in near future are fetched and stored in proxy server cache. Various data mining techniques can be used to accomplish the prediction tasks. Some of the approaches are listed below

* Association Rules
* Clustering
* Markov Prediction

**3.1 Association Rules:**

Association rules is one of the basic data mining techniques that is used to find associations among web pages that frequently appear in user sessions. Using web server access logs, groups of web pages are created where each group represents the pages accessed within a session. Rules are constructed based on pages that frequently appear together. An example rule would be of the form [ A ⇔ B, probability = 0.8 ] which indicates that if an user accesses page A, there is high probability (0.8) that page B would be accessed in the same session. Based on these rules, pages with high probability of being accessed are prefetched.

One of the biggest limitations of this approach is that order in which pages are accessed is not considered for constructing the rules. This is very important with web pages since accessing page-B after page-A is not same as accessing page-A after page-B.

**3.2 Clustering:**

Clustering techniques look for similar items based on the distance between them. In one implementation using clustering, user requests are used to construct a navigational graph (fig 3.1) in which each node represents a web page and each edge represents the frequency of transition between the 2 nodes (web pages). Clustering techniques are then applied on this graph. Each time a user requests a web page, all the pages which are in the same cluster as the requested page are prefetched.

One of the major limitations with clustering approaches is that as the number of web pages in the graph increases, they frequently become unmanageable.

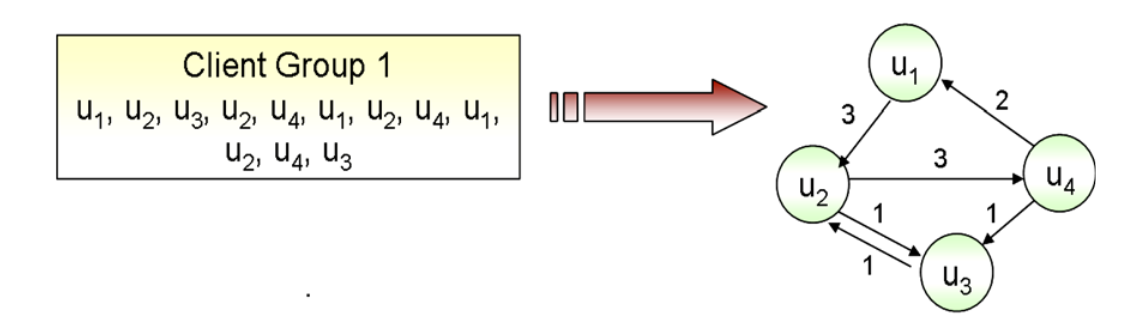


Fig 3.1: Navigational Graph

**3.3 Markov Prediction:**

Markov models are used in identification of patterns based on sequence of previously accessed items. Various implementations of markov models exist. Markov property states that, if present state is known then we can predict the future states, irrespective of what the past states were. To put it in other words if present state is known then future and past states are independent.We are using the K th -order Markov model as an approach for predictive prefetching, which computes the predictions by looking at the last K actions performed by the user, leading to a state-space that contains all possible sequences of K actions. But low order Markov models lack accuracy as their models do not use enough history. On the other hand, high order Markov models incur high state space complexity thus involving larger cost.

As a result of limitations mentioned with the techniques above, few variations involving a combination of data mining techniques were also proposed. 2 of them are outlined in the next 2 sections

**3.4 Proxy Side Predictive Prefetching Engine (PPPE)**

Since a proxy server is a [server](https://en.wikipedia.org/wiki/Server_(computing)) that acts as an [intermediary](https://en.wikipedia.org/wiki/Intermediary) for requests from [clients](https://en.wikipedia.org/wiki/Client_(computing)) seeking resources from other servers.A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource available from a different server and the proxy server evaluates the request as a way to simplify and control its complexity. A proxy server is a tool that can be suitably employed to reduce the WWW latency i.e. it can intercepts all requests to the web server to see if it can fulfil the requests by itself. If not, then only it may forward the request to the web server.

In fact, the proxy servers achieve two main purposes:

* Reduce latency: A Proxy server saves the results of all the requests from various clients for a certain amount of time. For Example, consider a case where both users A and B access the www through a proxy server. Let us assume that user A requests for a certain web page say page 5. After some time, user B also requests the same page. Instead of forwarding the request to the web server where page 5 actually resides, which can be a time-consuming operation, the proxy server simply returns this page from its cache where all the downloaded pages are retained before being overwritten by new arrivals. Since proxy server is often on the same network as the user, this is a much faster operation, thereby reducing the latency to some extent.
* Filter unwanted Requests: Proxy servers can also be used to filter the unwanted requests. For example, a company might use a proxy server to prevent its employees from accessing a specific set of Web sites. The www latency can be further reduced if the behavior of the user can be predicted and the predicted pages are prefetched accordingly and stored temporarily in the cache of the proxy server. As soon as the user asks for a page, the request can be fulfilled if the requested page is available in the cache. Considering this added advantage of proxy server, a prediction engine called Predictive Prefetching Engine (PPE) has been proposed that resides on the proxy server and processes the past references to deduce the probability of future access for the documents accessed so far.

The complete framework of Proxy side Predictive Prefetching Engine (PPE) shown in Fig 3.2

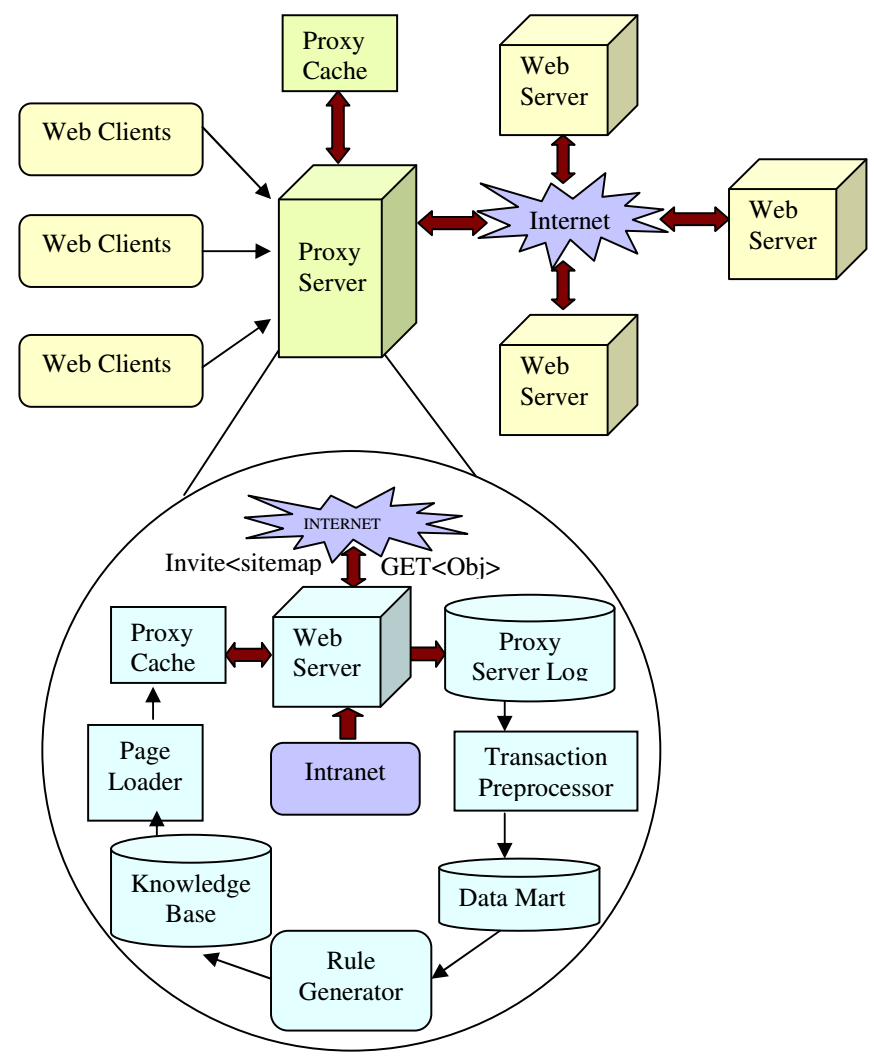


Fig 3.2: Framework for proxy side prefetching engine (PPPE)

Proxy side Predictive Prefetching Engine has the following components and processes:

1. **Proxy server log:** The records of all the users/clients that send requests to the server are kept on the web logs which are used to form the mineable warehouse. These warehouses are used to track the user activity. Since a proxy server sits between the client and the web server, it is comparatively easy to manage than the web server logs when it has to maintain the log for limited clients.

2. **Transaction Preprocessor:** Since in this framework, the log is being maintained on the proxy side, the transaction preprocessor operates on these proxy logs to accomplish preprocessing. Preprocessing involves the following steps:

* **Reduction of Search Space:** The first task is to reduce the search space for mining which is done by cleaning the proxy log for any unwanted records. It may include clearing the log from the irrelevant items like the image files (GIF and JPEG) and javascript files (JS) etc as these do not contribute for the patterns relevance.

Fig 3.3 below shows a sample web log before pre processing and Fig 3.4 shows the sample web log after pre processing

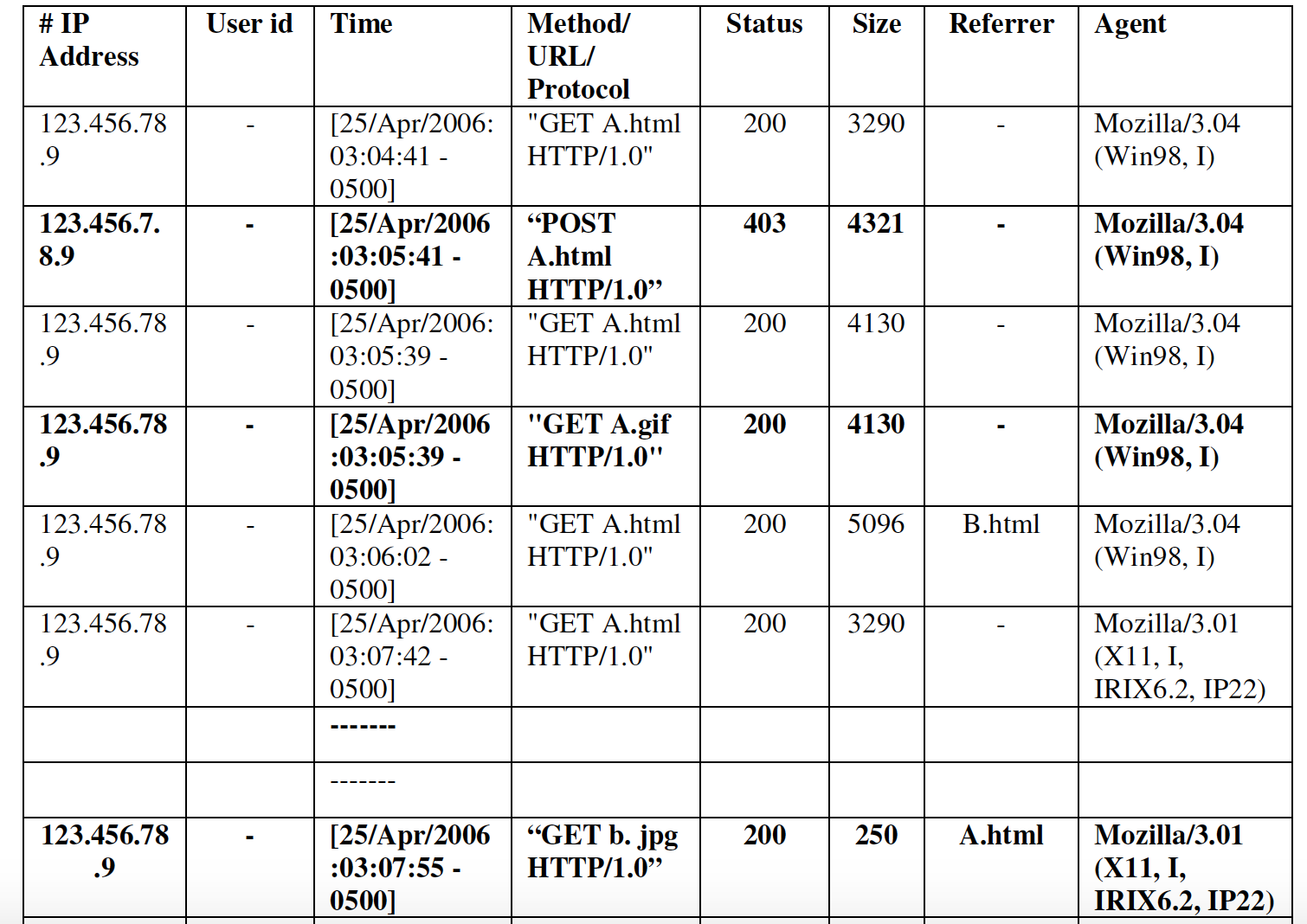
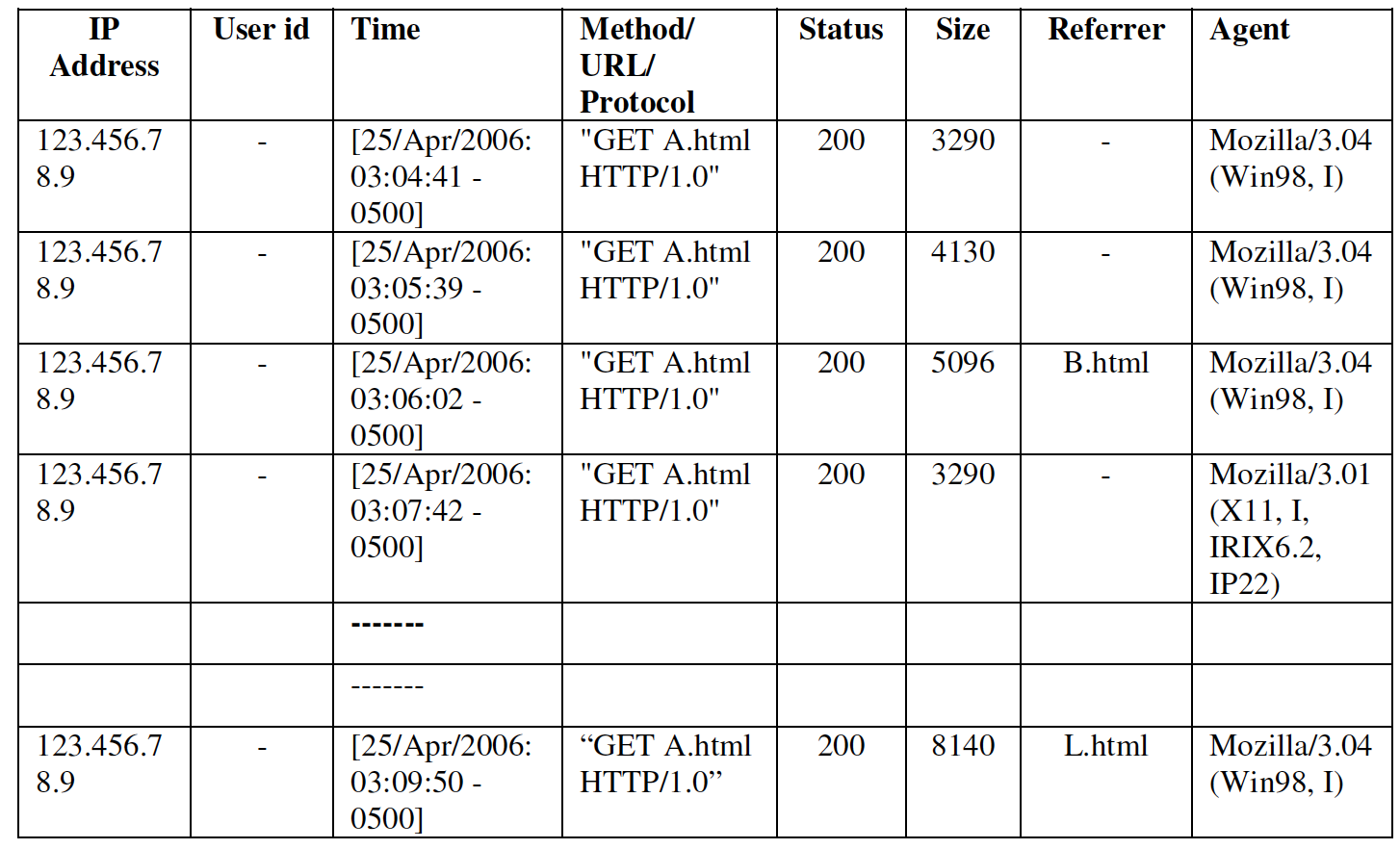


Fig 3.3: Sample WebLog before Preprocessing



* **User and Session Identification:** A user session is defined as the sequence of requests made by the single end user during a visit to a particular site. Within a single session, a user may follow links to several pages that belong to the similar pattern but during the same session, it may also be possible that the user might visit some other pages that do not belong to the same pattern. The user’s IP address is not sufficient for identifying a user. Many users can be assigned same IP address. Also many users can have access to the same computer. Cookies can be used for better user identification but are not brought into use due to privacy reasons. In such cases, User IDs are brought into use. The following criteria is used to identify the users:

1. If two records have different IP address, they are distinguished as two different users else if they have same IP address, then the user agent field is checked.

2. If the browser and operating system information in the user agent field is different in two records then they are identified as different users.

The session identification aims to divide the page accesses of each user into individual sessions. There are several heuristics available, According to that if the time difference between the two accesses, even from the same IP, is more than 30 minutes, then it is treated as a new session. Also, if the page stay time gap determined from the timestamp field exceeds 10 minutes, then also it is treated as a new session.

After all the preprocessing is done, the cleaner version of the proxy log is formed called Data mart.

3. **Data Mart:** Data mart acts as a database on which various data mining operations operate for generating the rules.

4. **Rule Generator:** It extracts the information from the data mart and applies the various data mining operations like association rules, clustering, markov predictors etc. to generate the rules for prediction.

5. **Knowledge Base (KB):** Knowledge base is a repository.The various rules generated by the rule generator is formed as the part of this repository.

6. **Page Loader:** For a given request made by the user, page loader consults the rules of the knowledge base and if the user’s requested pages exist in the heads of the rules, then the pages present in the body of those particular rules are prefetched. For instance, the kth entry in the knowledge base may have the following format:

Rk : Di > Dj ; if document Di has been requested then prefetch document Dj.

Similarly, nth entry in the knowledge base may have the following format:

Rn : Dj > Dk ; if document Dj has been fetched then prefetch document Dk also.

This method follows to the knowledge base till the time no more rules can be fired. To prevent increase in the network traffic, all the prefetched documents are stored in the data structure maintained in proxy cache as the Hint list.

**3.5 Search Engine Side Predictive Prefetching Engine (SPPE)**

In general, a common web search engine architecture consists of a front-end process and a back-end process. In the front-end process, the user enters the keywords in the search engine interface, which is usually a web page with an input box. The application then parses the search request into a form that the search engine can understand, and then the search engine executes the search operation on the index files.

After a proper ranking, the search engine interface returns the search results to the user. In the back-end process, a spider or robot fetches the Web pages from the Internet, and then the indexing subsystem parses the Web pages and stores them in the index files. The search engine retrieves the web pages according to the user query. Since relevancy is a subjective term, the search results may have varying degree of relevancy for different set of users. As a fact, there is an opportunity to significantly improve the relevancy of search results for a well defined set of users (example, employees of the same organisation), whose search habits are largely homogenous.

Predictive Prefetching Engine (PPE) sits behind the search engine interface. The intent of introducing the PPE is that it will increase the relevancy of the pages returned by the search engine according to the demand of the particular set of users which are termed as group clients. PPE also prefetches the pages if it lies in the rule-database that is generated by applying the various data mining operations on the group-client-log. This log is maintained by the search engine on the request of the various organisations which are assigned a particular set of IP addresses by the Internet Service Providers.

The interaction of the PPE with the user is explained with the complete framework as given in below Fig 3.5

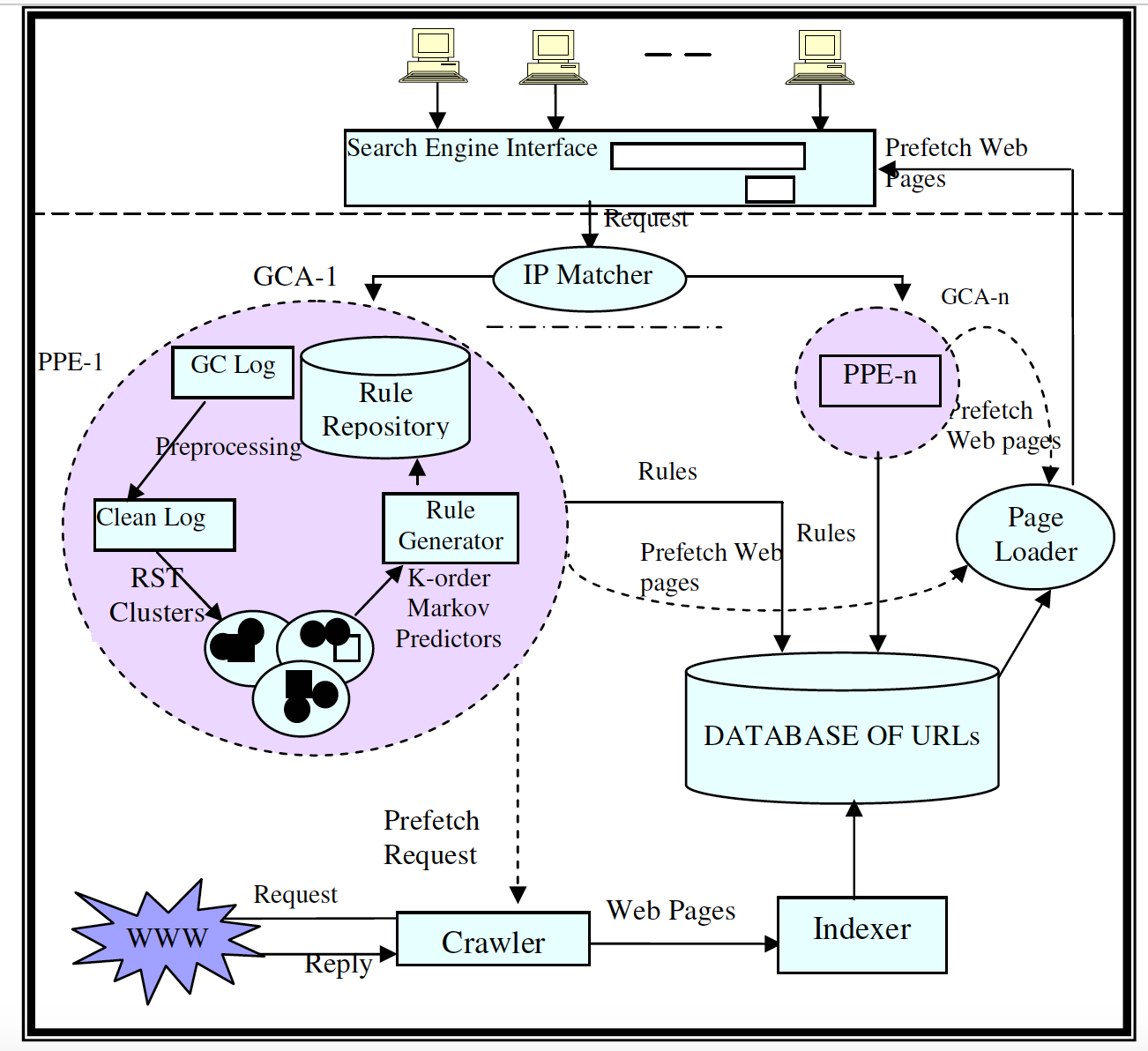


Fig 3.5 Framework for Search Engine side predictive Prefetching Engine (SPPE)

The various components of SPPE are as:

1. **Search Engine Interface:** It is basically a web page with the input box that is a part of the search engine’s front end and. The user enters his/her query that contains the keywords into the input box and hits the search button.

2. **IP Matcher**: IP matcher task is to extract the IP address from the query sent by a particular user. This IP address is then matched with the particular range of IP addresses for which different Group-Client-Agents (GCAs) are defined. Once the GCA is identified, it gets activated.

3. **Group-Client-Agent (GCA):** GCA plays an important role as it works on PPE. There are n GCA’s for n group-clients and therefore each GCA has its own corresponding PPE to work upon. One group client refers to a group of users within one organisation. Every organisation is assigned a unique set of IP addresses. Those IP addresses now forms a part of one group-client.

4. **Group-Client-Log(GC-Log):** This is maintained by the search engine on the Group-client’s request. The format of the log contains every entry from that particular group-client. Each record in the log file contains the client’s IP address, the date and time of the request that is received, the requested object and some additional information such as protocol of the request, size of the object etc.

5. **Clean Log:** A more cleaner version of the GC-log obtained by removing all the image files like .jpg and .gif from the GC-Log as they do not produce any productive information about the path followed by the user in a particular session.

6.  **RST Clusters:** A clustering technique is then operated on the clean log known as Rough Set Clustering. Clustering the sessions is to reduce the search space for applying the various data mining operations. RST operates on the principle of indiscernibility (two things are identical if and only if they share all the same properties). RST is chosen as the clustering technique as it helps in decision making in the presence of uncertainty. The result of applying RST is the lower approximation set (as shown in Fig 3.6) which contains all the user sessions which definitely has the target set.

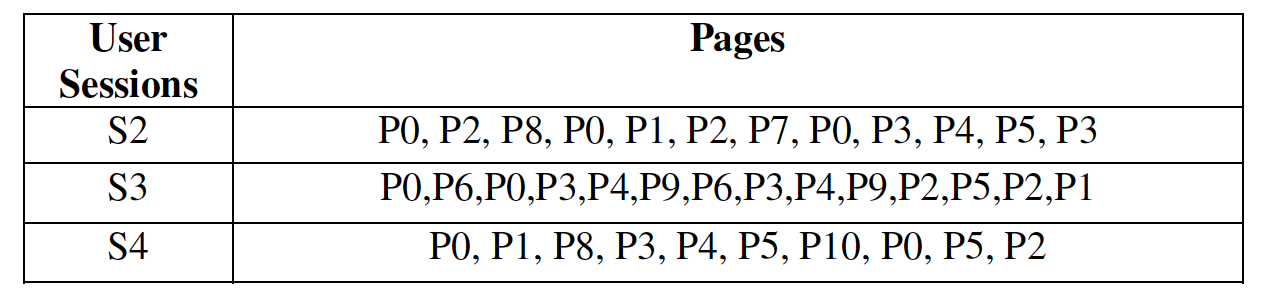


Fig 3.6: Lower Approximation set after application of RST

7. **Rule Generator:**  This phase takes lower approximation set as input and applies the k-order markov predictors (assumes the next term of the movement history depends only on the most recent k items). The order k is chosen dynamically every time depending on the size of the lower approximation set. The output is the rules of the form Di > Dj.

8. **Rule Repository:** The rules formed in the rule generator phase are then stored in this rule repository.

9. **Database:** The database contains the URLs of all the pages whose references are stored in the rule repository. The database is enriched by the URLs of rules from all the n PPE’s.

10. **Page loader:** Its task is to prefetch the pages populated in the hint-list by the GCA onto the client’s cache

**3.6 Evaluation of Predictive Web Prefetching:**

**3.6.1 Client/Server:** Prefetching is mostly at Server side since it fetches the predicted web pages from web servers and loads them into the proxy server’s cache while the user may be busy in performing some other tasks.

**3.6.2 Effect on Bandwidth:** The search engine retrieves the web pages for the general user. Since relevancy is a subjective term, the search results may have different degree of relevancy for different set of users. The work represents the PPE for retrieving the web pages for the particular set of users named group clients whose surfing pattern is logged in the CG-log maintained by the search engine only. Since, these group-clients reflects a particular behaviour over a period of time, PPE caches the same to return not only the relevant web pages but also prefetches them according to their history. Thus, PPE while prefetching the web pages makes sure that the network bandwidth is not wasted.

**3.6.3 Accuracy:** Markov model implementations have been hindered due to the fact that low order Markov models lack accuracy as their models do not use enough history. On the other hand, high order Markov models incur high state space complexity thus involving larger cost

**3.6.4 Timeliness:** In reality, large amounts of data grow steadily, some old association rules can become stale, and new databases may give rise to some implicitly valid patterns or rules. Hence updating rules or patterns is also important.

**Chapter 4**

**Data prefetching**

The aim of this prefetching is to reduce the time a person visiting a website spends on waiting for the requested resources. According to the data prefetching approach, the documents requested by the user may be directly connected to each other or share similar topic. One approach uses HTML hyperlinks as relations between the documents and their content to identify the subject. It works as follows :

* Whenever a user visits a web page, all the outbound links are collected with the surrounding text. Taking one link at a time from the collection, it’s content is analyzed as it is thought to describe the subject of the related links.
* The document is then cleaned from unnecessary HTML tags, punctuation marks as well as the common words which are not meaningful, in order to extract only relevant information.
* Next step is stemming which extracts the informative context by word suffixes removal.
* Based on the frequency of the terms, the terms appearing more are usually of less importance and the terms that are less frequent have high priority, so the least informative terms are eliminated to reduce the dimensionality.
* Applying the above procedure helps to describe the documents in terms of their content by using the context of links pointing to them.

Each user has a separate profile, which is used to store the keywords that the user finds interesting. Documents are considered to be interesting for a particular user, if the user has already visited it. Prefetching is done after applying weights to each outbound link on a subpage. These weights are calculated based on the particular link keywords and user preferences. HTML records are extracted and sorted based on timestamp. From these sorted access requests, HTML response set is built. For example,if we have responses d1; d2; d3; d4; d5 as the first set, we attempt to predict d6 using those. Likewise, d2; d3; d4; d5; d6 corresponded to the second set, from which we would attempt to predict d7 and so on.

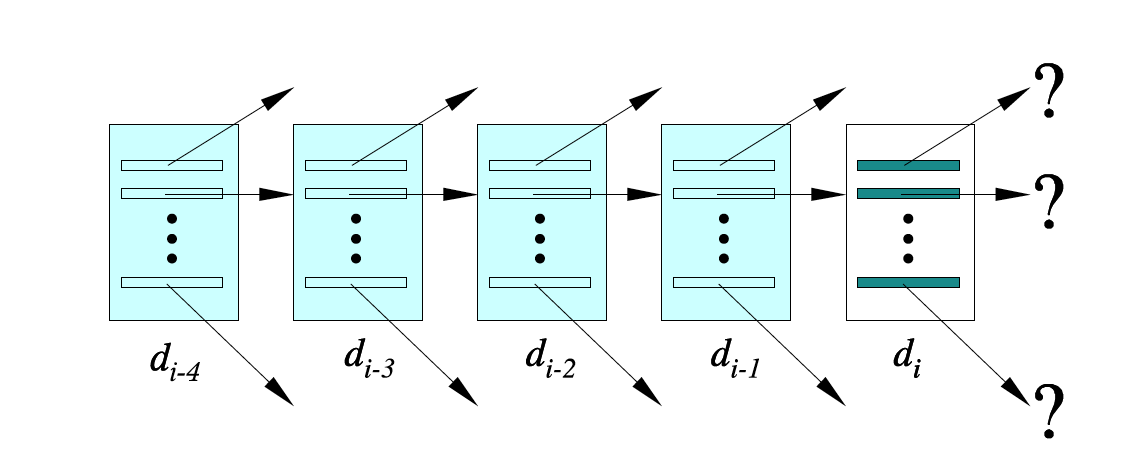


Figure 4.1 : Sequence of HTML documents requested. The content of recently requested pages are used as a model of the user's interest to rank the links of the current page to determine what to prefetch.

**4.1 Prediction method :**

There are four methods to rank the URLs of the current page for prediction. Two are simple methods: the baseline random ordering, and original rank ordering (i.e., first-occurrence order in page) . The second two methods, rank each link based on the similarity of the link text with the combined non-HTML text of the preceding pages (as illustrated in Figure 4.1). To measure the similarity between two text documents (D1;D2), a very simple metric is used:

TF(w,Di) = number of times term w appears in Di

Text-Sim(D1,D2) = TF(w,D1)\* TF(w,D2)

Since links are typically presented within some context, in addition to anchor text the terms before and after the anchor are also considered. For example, if the HTML content of a page were:

The <a href="http://www.tajmahal.gov.in/">Taj Mahal</a> is one of the wonders of the world. Then 0 additional terms would provide just {Taj Mahal}. If 10 additional terms are permitted on each side, then we get the set {The,Taj Mahal, is, one, of, the, wonders, of, the, world.} to be used for comparison against the text of the preceding pages. This set can be filtered and {the,is ,of} words of less importance can be removed to get a refined set of key terms like {Taj Mahal,wonders,world}.Similarly by permitting 5,10 or 20 additional terms results can be improved.

Example : Technique used by Google for their Picasa Web Album - Image files take longer to download then most kinds of files on the web. When a user looks at a photo on Picasa Web Albums, they make a guess that the user will look at the next photo as well, and start downloading it as soon as possible,making the next photo available almost instantly.

**4.2 Data prefetching in Mobile service application:**

In mobile computing, one of the most important parameter of the user perceived QoE is the round-trip-time between a sender and a receiver. In many mobile application scenarios, certain data is required to be provided in a timely manner such as:

* Audio data from a music streaming service will cause playback stutter when the device-side buffer underruns
* a turn-by-turn navigation system must provide users with on-time, detailed graphical instructions, possibly generated dynamically by a centralized service in order to reduce computational load of the client devices
* live traffic data can be provided to the client in real-time in order to provide the user with dynamic rerouting in case of traffic jams. Also, an increase of applications deployed on mobile devices but using Web- and Cloud-based resources can be observed.
* This shift towards mobile computing calls for new ways to improve the QoE for mobile users:
  + Mobile devices are often connected to the Internet using mobile data networks like 3G (UMTS, HSDPA/HSUPA) or 4G (LTE), which are by nature subject to fluctuations and instabilities.
  + Some of these variations are accountable to the provider (e.g., caused by over-provisioning), but often it is the location of the device which leads to low connection quality, e.g., because the user is driving through a tunnel or simply since the network coverage in rural areas is inferior to urban environments.
  + Furthermore, it is noteworthy that location influences connection quality not only because of geographical features (hills, tunnels, remote areas), but also because some locations might indirectly cause bad connection

**4.3 Evaluation of Data Prefetching:**

**Performance:** Data prefetching proves better for mobile devices but, it is noteworthy that all prefetching approaches, deployed to mobile clients, bear the danger of creating ripple effects: a high amount of clients, prefetching data during times of good network quality, actually decrease the network quality during this time. Deploying such mechanisms on large-scale networks requires appropriate mechanisms for preventing such situations.

**Latency :** The above mentioned algorithm claims reduction of latency from 25% up to 75%, depending on the number of prefetched documents, but with the cost of increasing the bandwidth overhead.

**User behaviour modeling:** The problem of user behaviour modeling for prefetching is that it is just the tip of the iceberg. There are many related problems, like adaptive interfaces, recommendation engines and content personalization which may directly result in a revenue increase of the provider’s business.

**Cost and accuracy :** Due to high bandwidth overhead the overall cost is high.

**Chapter 5**

**Semantic Web Prefetching**

This techniques uses semantic web for prefetching analysis. It is based on semantic locality.

It is implemented using following components.

1. Lexical Analyzer
2. Predictor
3. Prefetcher
4. Storage Unit

The following diagram shows the arrangement of the blocks.

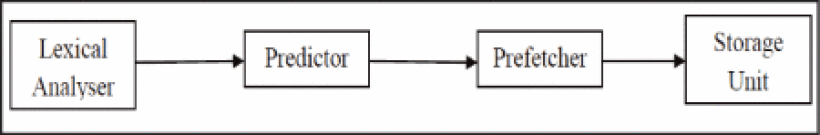
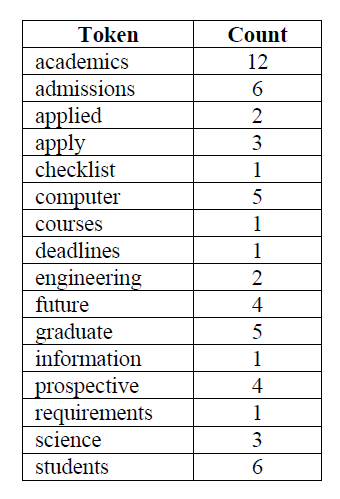


Fig 5.1 Semantic Web Prefetching Block Diagram

**5.1 Lexical Analyzer**

It is also known as Tokenizer. This unit parses the entire web page, it extracts anchor links. The anchor text is then converted into tokens, which consists of keywords. The tokens are added to a list which consists of key value pairs. For each token, the frequency of occurrence in the current web page is recorded. Every time, a new occurrence of a particular token is found, the frequency value is incremented. This list is given as input to the predictor unit.



**5.2 Predictor Unit**

The predictor unit gets the list as input, it calculates probability of each token and provides a new list of links that are likely to be requested by the user in near future.

There are various techniques available to predict the next page. These techniques use various statistical, stochastic methods. In additions, different parameters like thresholds, information gain are considered to improve the accuracy of results.

**5.2.1 Decision Tree Induction** - In this case, probability of each token is calculated. It is followed by calculation of information gain value of each token. Based on this value, the decision tree is formed. Some pruning could be done to improve performance of the decision tree technique.

The list of links or web pages to be prefetched is compiled or generated based on results of predictor unit.

**5.2.2 Naive Bayes method** - The predictor can also be built using Naive Bayes Classifier. It is simple, easy to implement. It requires less computation and lower storage space as compared to Decision tree or other techniques.

The Naive Bayes calculates probability of each of each token and produces a list of web pages to be prefetched based on non-decreasing probability of the pages.

Following diagrams show comparison between SPRINT, which is a decision tree induction technique and Naive Bayes.

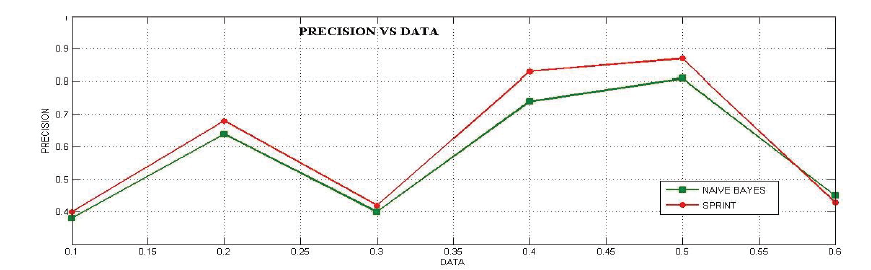


Fig 5.2 Precision vs Data

As Sprint involves more complex calculations and it considers many additional attributes, it is more precise than Naive Bayes.

**5.3 Prefetcher**

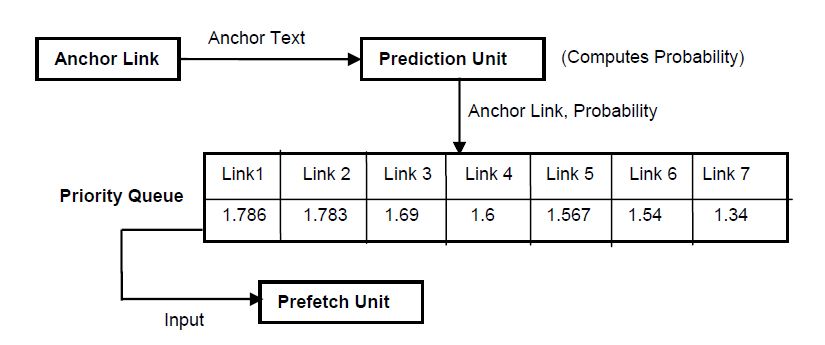


Fig 5.3 Prefetch Unit

The output of predictor unit, which is a list, is given to the prefetcher. the prefetcher prefetches the web pages when the browser is idle and not responding to any particular request from the user.

The web pages are chosen based on threshold and cache limit of the client machine.

It is important to keep in mind that the overall user experience should not get hampered due to prefetching actions in background.

**5.4 Storage Unit**

The client side cache is used to store the web pages that are prefetched by the prefetcher unit. The allocated space for this storage is limit and hence, page replacement algorithms are required. It is customary to use Optimal Page replacement algorithm as the future requests are already known, allowing the lowest fault rate.

This storage unit is separate from the browser cache storage.

Following steps show the summary of semantic web prefetching.

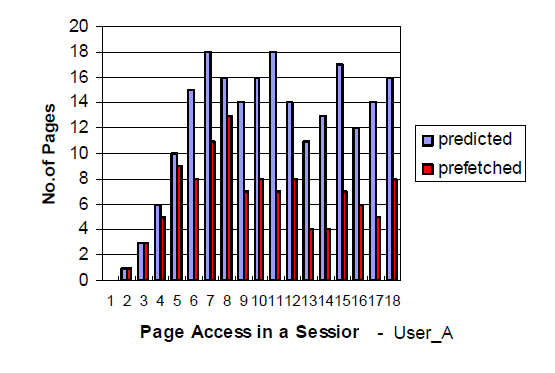
Steps involved in Semantic web prefetching

1. User requests a URL
2. The associated web page is served by web server and displayed by the web browser.
3. The web page is parsed by the tokenizer / lexical analyzer
4. Anchor text is extracted from each anchor text associated with anchor link.
5. If the anchor text exists in the repository, the frequency value is incremented, else new entry is added.
6. The probability values are calculated and prefetch list is prepared.
7. The web pages in the prefetch list are prefetched when the browser is idle.
8. These prefetched web objects are stored in client cache.
9. If the user requests a web page from the prefetched list, page is displayed from the cache.
10. Else it is requested from the web server.

**5.5 Analysis of Semantic Web Prefetching**

Below, is an analysis of semantic prefetching for two users, UserA and User B. User A is searching for a specific topic, for instance Computer Science Universities in Bay Area. User B on the other hand, is browsing the web without any specific intent in mind.

SEmantic web prefetching performs better for User A, it provides more number of web pages in the predicted list. It also has a better hit rate compared to User B.



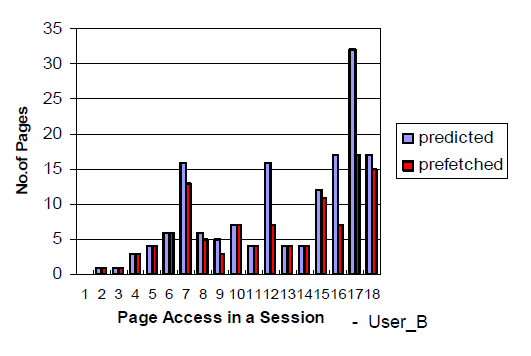


Fig 5.4 Page Session Analysis

It can be stated that, semantic web prefetching is ideal for text based website. It is suitable and gives more accurate results when user is looking for a specific topic. In addition to that, the results improve as the length of time the users spends in a session increases.

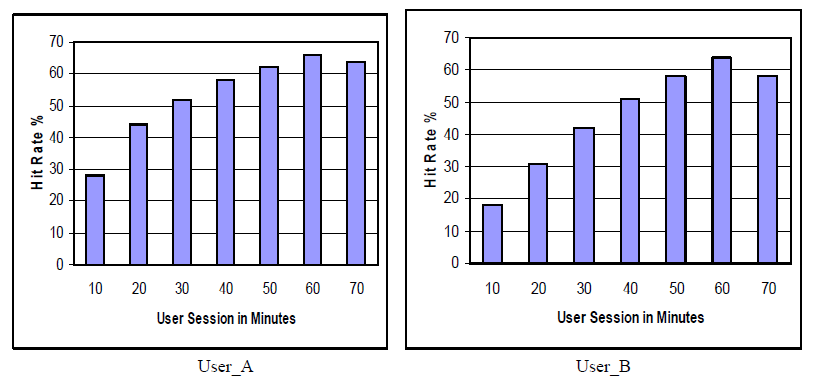


Fig 5.5 User session time analysis

Another observation about this technique is that the performance improves as the time a user spends in a particular session increases i.e. longer the user session, better the results.

Above diagram shows that even for User B, the hit rate % increases as he spends more time.

**Chapter 6**

**Design considerations**

#### While designing a good prefetching technique, it is important to understand the following factors:

#### **Study of user actions -** If the users spend most of their time on one particular page or action, then that's the one to optimize. It can be figured out by looking through server logs, or just by watching a few volunteers use the site. This can help to build the pattern of web browsing for different types of users.

#### **User device considerations -** In addition to user behaviour, it is necessary to take into account type of devices, bandwidth, overall network strength of the users. It is ineffective to use client-intensive prefetching techniques if most of the users have mobile devices.

#### **Measuring when the page is ready -** Data for the new page should not be fetched before the data for the current page is done. For very dynamic pages, JavaScript onload handlers may be needed for each external file on the page. Once those resources are safely downloaded, it's reasonable to start prefetching files for the next user action.

#### **Prefetch the right data -** Some data is safer to prefetch than others. Images tend to be long-lived, and cache very well in the browser. Many data feeds are safe to prefetch, while others may be time sensitive (for example, a feed of recent updates), or frequently modified by user actions to be good prefetching candidates.

#### **Browser Dependency -** It should be kept in mind that some prefetching techniques are browser dependent. The implementation, type of data structure, cache limit, threshold, computation methods used in a technique might differ across browsers affecting the performance and efficiency of the prefetching results.

#### **Profile the changes -** Using profiling tools like Page Speed to get speed measurements before and after any change is made is a good practice. Making sure that the change is not making the rest of site slower by requesting too much data at once. Using tools to tune how much and how frequently prefetching is done, based on how much user time is saved compared to how much network bandwidth is used.

#### **Computational competence -** Keeping in mind the computational complexity and the data requirements while choosing a proxy technique for prefetching is very critical. For techniques using Naive Bayes, the computation and storage needs are minimal, but the results may not always be satisfactory. On the other hand, methods like Decision Tree induction, Markov predictions provide better results, but require more complex and data-intensive computations.

#### **Be a good web citizen -** The user has many web sites open at a time. It is important to be cautious about the amount of bandwidth used for prefetching a particular web site. The overall experience should not be hampered.

**Chapter 7**

**Conclusion**

Today’s users want seamless, fast and efficient web experience. The web pages are filled with dynamic content and media. Web prefetching helps in giving users the experience they need and deserve. It is a technique providing low latency and better performance.

Web prefetching involves predicting the future request of the users based on their current and past browsing behaviour. There are various techniques used depending on the type of website, type of user, overall bandwidth available. In addition to that, storage capacity, computational complexity also affect the choice and in turn efficiency of prefetching technique used.

As the web experience becomes richer, more multimedia intensive and more interactive, further research will take place in the area of web prefetching. Better and more efficient techniques will be implemented improving overall user journey on Internet.

**References**

1. A study on web prefetching techniques (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.471.7383&rep=rep1&type=pdf)
2. DDG: An Efficient Prefetching Algorithm for Current Web Generation - http://ieeexplore.ieee.org/document/4178377/
3. https://hostingfacts.com/internet-facts-stats-2016/
4. http://www.technewstoday.com/26073-twitter-and-google-to-release-faster-web-browsing-feature-for-mobile-users/
5. Achuthsankar S. Nair & J. S. Jaya- sudha,(2007) “Dynamic Web Pre-fetching Technique for Latency Reduction”, International Conference on Computational Intelligence and Multimedia Applications.
6. Darin Fisher & Gagan Saksena, (2003)“Link Prefetching in Mozilla : A Server-driven Approach”, Proceedings of 8th International Workshop on Web Content Caching and Distribution.
7. V. N. Padmanabhan; J. C. Mogul. “Using Predictive Prefetching to Improve World Wide Web Latency,” Proceedings ACM Conference, SIGCOMM, 2008.
8. A. Georgakis, H. Li, “User behavior modeling and content based speculative web page prefetching”, Data & Knowledge Engineering 59 (2006) 770–788
9. Robert Cooksey. “Content-Sensitive Data Prefetching”, A thesis submitted to the University of Colorado requirements for the degree of Doctor of Philosophy, 2002.
10. Cheng-Zhong Xu; Tamer I. Ibrahim. “A Keyword-Based Semantic Prefetching Approach in Internet News Services”, Journal of IEEE Transactions on Knowledge and Data Engineering, Vol. 16, Issue 5, 2004.
11. https://www.maxcdn.com/one/visual-glossary/prefetching
12. https://en.wikipedia.org/wiki/Link\_prefetching#cite\_note-12
13. http://s3.amazonaws.com/academia.edu.documents/32049506/J5.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1489970788&Signature=6q0Ib2FzWpbxKpcgeGHVnYMcLkQ%3D&response-content-disposition=inline%3B%20filename%3DA\_clustering-based\_prefetching\_scheme\_on.pdf
14. https://blog.stackpath.com/glossary/lazy-loading/
15. https://developer.mozilla.org/en-US/docs/Web/HTTP/Link\_prefetching\_FAQ
16. https://css-tricks.com/prefetching-preloading-prebrowsing/
17. Brian Douglas Davison, “The Design and evaluation of web prefetching and caching techniques.”