Journey to the Past: Proposal of a Framework for Past Web Browser

Adam Jatowt[1], Yukiko Kawai[2], Satoshi Nakamura[1], Yutaka Kidawara[3] and Katsumi Tanaka[1]

[1]Kyoto University
Yoshida-Honmachi, Sakyo-ku,
606-8501 Kyoto, Japan
Phone: +81-75-7535969
{adam, nakamura, tanaka}@dl.kuis.kyoto-u.ac.jp

[2]Kyoto Sangyo University
Motoyama, Kamigamo, Kita-Ku,
603-8555 Kyoto, Japan
Phone: +81-75-7052958
kawai@cc.kyoto-su.ac.jp

[3]National Institute of Information and Communications Technology
3-5 Hikaridai, Seikacho, Sorakugun,
619-0289 Kyoto, Japan
Phone: +81-77-4986828
kidawara@nict.go.jp

ABSTRACT

While the Internet community recognized early on the need to store and preserve past content of the Web for future use, the tools developed so far for retrieving information from Web archives are still difficult to use and far less efficient than those developed for the "live Web." We expect that future information retrieval systems will utilize both the "live" and "past Web" and have thus developed a general framework for a past Web browser. A browser built using this framework would be a client-side system that downloads, in real time, past page versions from Web archives for their customized presentation. It would use passive browsing, change detection and change animation to provide a smooth and satisfactory browsing experience. We propose a meta-archive approach for increasing the coverage of past Web pages and for providing a unified interface to the past Web. Finally, we introduce query-based and localized approaches for filtered browsing that enhance and speed up browsing and information retrieval from Web archives.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *navigation*

General Terms

Algorithms

Keywords

past Web browser, Web archive, past Web

1. INTRODUCTION

The World Wide Web changes continuously, generally in an unpredictable and unorganized manner, and it is necessary to store the past contents of Web pages for future reuse. The Internet Archive [19] is the best-known, general Web archive; moreover, currently it is probably the largest digital library in the world. It offers about one petabyte of data and is growing at a rate of 20 terabytes per month [19]. Other dedicated Web archives also exist such as archives containing Web pages from individual countries [1,2,17,24], for example: Australian archive [24], Swedish archive [2] or thematic Web archives related to certain events or topics, such as: September 11 Web archive [25] or Election 2002 Web archive [14]. Besides these, there are other repositories of past Web pages that can be considered to some extent as Web archives, such as: local caches, site archives, personal Web document repositories, transaction-time servers [12,13] or search engine caches. Lastly, there are organizations like International Internet Preservation Consortium [18] that were established with a mission of collecting, preserving and making accessible data from the Web for future generations.

Web archives show the history and evolution of the Web as well as reflect the past states of societies. They contain evidences of how our society has evolved and how it reacted to events in the past. Historical snapshots of Web pages can potentially show the history of the underlying elements represented by those pages—institutions, companies, people, and other such entities. The usage of Web archives can greatly benefit researchers and practitioners in many areas ranging from history to sociology and marketing. Recently some concerns arouse, however, with copyright issues of data stored in Web archives. To cope with this problem, Web archives introduce policies to allow content publishers to choose whether they consent to having their content archived.

Throughout this paper, as the live Web we consider the present Web containing Web pages as we see them currently. These pages have potential of being updated, hence, we call them live. Past Web, on the other hand, is considered as the part of WWW space where pages have no longer any change potential and are merely snapshots of past states of live pages. Hence, past pages are the frozen snapshots of pages that once belonged to the live Web.

Since much of the information that has disappeared from the live Web is still available in Web archives, it is reasonable to assume that future information retrieval systems will be able to retrieve data from the live and past Web at the same time. This will increase the total amount of information available. Users might search for some information from the past, which has already disappeared from the present Web and cannot be retrieved by conventional IR systems. While old page versions may not be fresh, nevertheless their contents may offer some value. They also have the advantage of guaranteed persistence in an unchanged form. Moreover, browsing past content of current, live pages can help to characterize them better than merely seeing their present momentary versions. For example, Web authors may gain more

HT'06, August 22-25, 2006, Odense, Denmark.

knowledge about a certain live page and utilize that knowledge when deciding whether or not to link to it. Thus, they will be more convinced about the general scope of topics of the page and the probability of its future topics. Additionally, many users may find it interesting to browse histories of their favorite Web pages to see how they looked like before.

Considering the advantages and the potential of Web archives, the realization of a browsing interface that enables users to find and to view past page versions in an easy and efficient way would be of great benefit. While there are various visualization interfaces for digital libraries, an interface similar to those for the live Web seems the most convenient for Web archives. The interface should support fast and easy information retrieval and browsing. Ideally, it should offer access to the past Web that is similar to and as efficient as that offered by current interfaces to the live Web. However, current interfaces to Web archives are still in the development phase and do not necessarily provide such access. Browsing the past Web is currently not efficient due to three problems in particular.

- Difficulty of navigation in space and time in Web archives.
- Lack of efficient change management, which would help to perceive the evolution of pages in time and help to find required information in massive amounts of past Web data.
- Limited coverage of Web archives, hence, lack of historical data of many Web pages or its incompleteness.

We have developed a general framework of a browser that should facilitate and enhance the browsing of Web archives. A browser based on our framework would support passive browsing with minimal user interaction. The user would input URL and a time point from when he or she wants to start browsing. The system would then sequentially retrieve past page versions from archives, starting from the specified time point, and present them to the user one by one. This presentation of page versions would be based on detection of changes and conversion to an animation. The user could then view the parts that had been changed along with the time flow. This approach makes it easy for the user to observe and to understand the evolution of pages in time and to detect the updated content.

We can distinguish two basic types of browsing: vertical and horizontal. The former means navigating between past versions of different pages at or around a certain point in time, while the latter means viewing page versions of the same page along the time direction, i.e., browsing the past Web in a horizontal direction. A mixture of both kinds of browsing would enable users to traverse both in time and space of the past Web. While we are particularly interested in the horizontal type of browsing, our framework incorporates a mixture of both types.

A major problem encountered when browsing Web archives is an incomplete state of past Web snapshots. Due to resource limitations, Web archives cannot crawl and preserve the whole Web with suitable granularity to capture all changes in Web documents. We propose a meta-archive browsing approach for providing a unified access to the past Web to overcome this problem. This approach assumes searching for and collecting past page versions from different past Web repositories in the same time.

Our proposed framework incorporates also methods for improved browsing when a user is searching for certain information. We provide query-driven and localized filtering capabilities, which are especially useful for information seeking users when the exact date or query is not known or is unclear. This information need can be reflected by a request, for example, such as "I would like to find information about X that appeared some time ago on page A." If the exact time point is known, the problem is trivial. However, people often do not remember when they saw some particular bit of information. Another type of query that can be effectively handled by filtered browsing is one about the date of a deletion of content, such as "I would like to know when information about X was removed from page A." For example, a user may want to know when a particular member of some laboratory left, so that his or her name was removed from the list of laboratory personnel[1].

The remainder of this paper is organized as follows. After first discussing related research in Section 2, we describe in Section 3 our proposed framework for browsing the past Web. In Section 4, we describe its search-oriented features. In Section 5, we discuss the implementation of the browser while in Section 6 we demonstrate the validity of our framework based on experimental results. In Section 7, we discuss how the assumptions and simplifications we made limit our framework. We also describe the contributions our framework makes and its potential applications. In the last section, we summarize the key points.

2. RELATED RESEARCH

The dynamics of the Web has been measured in a series of experiments in which vast quantities of Web pages were analyzed and compared over a certain period of time [4,6,15]. The results showed that the Web is a very dynamic environment in which changes continuously occur in both the content and the link topology. While, it has been revealed that many new pages are continually being added and many old pages are being deleted, there are still a large number of stable Web pages that have long histories, which could be browsed. The main or top pages of Web sites often fall into this category. Our primary focus is such pages.

Until now the Web archiving community has mostly concentrated on storing and preserving Web pages; relatively little attention has been paid to providing interfaces to the past Web data. The Wayback Machine [19] is a popular Web-based interface to the Internet Archive. It is more a general retrieval system than a traditional browsing application. After a user inputs a URL and optionally dates defining a requested time period, the available page versions of the URL are displayed in sequential order in the directory page. The user can click on any page to view its contents. He or she can follow links from that page if the linked pages are also stored in the archive. The Wayback Machine also indicates which page versions contain changes by marking them with an asterisk; however, it does not display changes unless the user explicitly requests change visualization. Generally, the Wayback Machine does not promote easy serendipitous discovery in the Internet Archive. Horizontal browsing along the time line is troublesome and takes time since the user needs each time to

[1] Although there would normally be a delay between the person's departure and its reflection on the page, if we assume that it is minimal, the date of removal should reasonably approximate the date of departure.

come back to the directory page and to click on a new page version. This becomes even more difficult when he or she wants to freely browse the spatio-temporal structure of the past Web. Recently, a public open source implementation of Wayback Machine has been also released [30]. Besides having most of the functionalities of the standard Web-based version, it provides Timline Replay mode in which a timeline listing page versions is visible at the top of each page. Browsing forward and backward in time is done by clicking on the timeline. However, again, this style of browsing requires users to click on page versions, which might be troublesome when there are many past page snapshots.

WERA (Web Archive Access) [31] is another archive viewing application. It was built from the Nordic Web Archive (NWA) Toolset [17,22], an interface to the Nordic Web Archive [17]—a joint project of the national libraries of the five Nordic countries. WERA is similar to the Timline Replay mode in the open source implementation of the Wayback Machine. It supports time and URL input for specifying a particular page version. The user interface also has a simple timeline showing the number of available page versions and the currently browsed page version. Similarly to the Wayback Machine there is no change visualization integrated, so identifying added and deleted content can be time consuming and error-prone, especially for Web documents with much content.

Both the Wayback Machine and WERA are designed for particular Web archives. This implies that only the resources stored in a single repository or in a small, limited set of archive collections can be accessed during browsing. However, individual repositories are always incomplete considering the expected size of the past Web. To overcome this limitation, our framework allows for the usage of virtually unlimited number of Web archiving repositories at the same time. Hence, we call it past Web browser rather than Web archive browser. Next, it facilitates browsing the past Web by combining passive, automatic page viewing together with change detection and presentation. This is very useful for efficiently browsing large amounts of past page versions that additionally could have few updated contents. We also provide tools to facilitate navigation in the link structure of the past Web. Finally, our framework has functions that minimize the user effort and time required to find specific information in past versions of pages.

As we mentioned before browsing of past Web data is done in our framework using the change detection and change presentation system that facilitates user comprehension of page history and helps to find requested content. Several automatic change detection and presentation systems have been proposed for online data [3,11,20]. However, these systems, including the one used in the Wayback Machine, are not efficient when there is a change overload and can fail to clearly show deleted and added parts of the pages, especially if their locations overlap. Presenting both the added and deleted parts on one page may result in too much data being blended together in a manner not suited to the page structure. Our framework avoids this problem by visualization of the changed elements using animation effects.

Offline browsing [23] bears some similarity to past Web browsing. Offline browsers download data from the Web so that it can be browsed later on local machines. In this way users can view previous snapshots of fast changing pages in detail at any time. However, browsing along time is not supported, nor other functionality proposed in our framework.

User navigation patterns and usability of navigation tools for the live Web have been studied in [7,26]. Unfortunately, to the best of our knowledge, there was no similar study conducted for browsing past Web page versions stored in Web archives.

Chi et al. [5] and Toyoda and Kitsuregawa [27,28] demonstrated systems for visualizing the evolution of Web ecologies or communities by analyzing and visualizing the snapshots of Web graphs. While these applications display the changes in the structure and popularity of parts of the Web, our focus is on visualizing changes in single Web documents and on providing a general tool to a user for unrestricted browsing in the past Web.

Finally, there is large research work that to varying extent is related to temporal aspects of the Web. For more information, [16] contains an exhaustive selection of research papers that deal with time in the context of Web pages.

3. PROPOSED FRAMEWORK

3.1 Meta-archive Approach

In the past Web history of each Web page, marked by a unique URL, can be represented using its past versions. Page version is a frozen snapshot of page contents at some point in time and can be considered as a sample drawn from the page history, reflecting the state of the page at a certain time point. The page version is thus marked by the URL of the page and a timestamp indicating the date of its capture. Note that under this definition, two page versions retrieved from different time points can still have the same content.

The representation of the history of a page is often incomplete. Consider two consecutive versions of a page v_1 and v_2, drawn randomly from a Web archive database. The time points when the page versions were fetched and stored in the database are t_1 and t_2 ($t_1<t_2$). The probability $P(v_i)$ that there is a v_i satisfying $t_1<t_i<t_2$ and containing content different from that in v_1 and v_2 depends mostly on the length of the period from t_1 to t_2, but also on such factors as: the type of the page, the content difference between page versions v_2 and v_1, the average changeability of the page, etc. The probability $P(v_i)$ will be coming closer to zero as the length of the period from t_1 to t_2 is decreasing, however, it could never reach zero as the page cannot be crawled continuously unless there is some other evidence of a lack of any unknown changes.

We conceptually define the coverage of a Web archive as the total amount of different content tracked and stored in the archive compared to the total amount of different content that ever existed in the Web before. Additionally, Web archives can be evaluated from the viewpoint of the trustfulness of the archived data they contain that could be evidenced in practice by some kind of certificates. For example, data obtained from a personal Web archive would be considered less trustful than data collected from a large Web page repository containing billions of page versions and having professional maintenance and control.

Due to resource limitations, Web archives often have very low coverage as they contain only few snapshots of tracked pages or they do not track some pages at all. Our proposed framework alleviates this problem by using a meta-archive browsing approach to maximize the overall coverage at low cost. This approach presumes communication with several complementary Web archives at the same time (Figure 1). The list of past Web repositories that can potentially contain searched data is stored and maintained by the browser. After receiving a request for a

page history, the system queries the archives about their versions of the page for the specified time frame. The optimum strategy would be first to fetch and to compare the document signatures (checksums) in order to detect those page versions that contain changes among the cooperating repositories. Downloading only such versions would result in maximum efficiency of fetching data for the reconstruction of page history. Alternatively, the system would require lists of those page versions from individual archives that have changed content when compared to the neighboring (consecutive) page versions stored in those archives. Note that the Internet Archive already provides lists of Web page versions indicating those that were changed. Then, only the versions with a change would be fetched from the archives. For other archives, which do not support such functionalities, all stored snapshots of the page would have to be downloaded. Page versions, represented by their URL addresses and timestamps would be sent to the browser, which then would place them in order and make ready for viewing. The browser could also compare Web archives' certificates to determine the levels of trustfulness and choose the versions with the highest levels. Since the entire interaction must be realized in real time, it is necessary to ensure an uninterrupted and rapid stream of data. Besides this, load-balancing measures could be taken to prevent too high workload of single resources.

The realization of the meta-archive approach would bring us closer to the experience of browsing the past Web rather than browsing individual past Web repositories. We would have a unified interface to the history of the Web and be less dependent on individual Web archives. However, since Web archive interfaces are still not standardized, the browser should currently support different communication protocols and data acquisition methods for different past Web repositories.

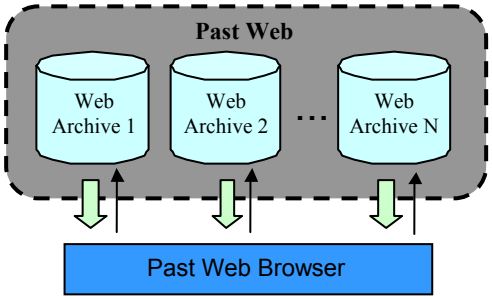

Figure 1 Meta-archive approach as an interface to the past Web

3.2 Horizontal Navigation

Browsing begins from a starting point, represented by a URL and a point in time. If there is no corresponding page version in the archive, the browser chooses the one that is closest in time. Next, upon pushing a play button, the browser automatically displays consecutive page versions in a passive way. Passive viewing in horizontal browsing results in minimum user interaction since page versions are presented to the user one by one, like in the slideshow. Similar to watching a video, the user can pause or stop the motion, enabling detailed viewing of the current page version or link following. In addition, at any time the user can manually input a new date or a new URL, thereby jumping to another page version or to another Web document.

The browser features a timeline that is automatically constructed when the presentation flow of a certain page begins. It shows the distribution of page versions and their changes. It thus displays the points in time for which page versions are available and a color indicator showing whether their content differs from that of previous versions. The currently viewed page version is also indicated in the timeline. The information provided by the timeline enables the user to orient himself or herself in the time dimension and to check whether further horizontal browsing would be worthwhile. The timeline is also a navigation tool as the user can click on any page version and jump in time. Zoom function is also provided to see the timeline in higher detail.

The construction of the timeline is a costly process since the browser has to fetch all page versions and to compare them for changes. Therefore, it is gradually constructed during browsing, starting from the page versions closest to the actual time point of navigation. This was however facilitated in our experiments since most of data was obtained from the Internet Archive, which provides information about the distribution of changes in page histories.

Additionally, a bookmarking facility is also provided. Unlike in the live Web, bookmarks in the past Web usually guarantee the preservation of a page version in an unchanged form. Users can select a page version and bookmark it for later use as a starting point for the next browsing experience.

Finally, we propose a recording function like in traditional video recording systems. It gives the user the ability to record the history of pages in a customized way and to share it with others. The browsing history is recorded as past page version indicators (URL, timestamp) and optionally user actions that were performed during the browsing. Other users who use the same browser could then replay the browsing again on their local machines. This could be useful when users wish to share information about interesting changes they noticed while browsing.

3.3 Change Detection and Presentation

Our approach is based on the belief that browsing of the past Web should be based on change management. Considering the current size of the past Web, where there is a lot of static and redundant data, we believe that the optimal method for browsing is the one using the change visualization. It enables us to see the essence of the history of Web documents and to view their evolution. We thus assume that the changed data is the most important element in Web archives and that focusing on it can help reduce the amount of browsing in the massive data pools of Web archives.

Added as well as deleted types of content changes between sequential page versions are displayed to show the overall content variance in the page. This enables users to spot not only the added content in consecutive page versions but also to observe the removed content.

Our change presentation algorithm assumes the gradual display of changes in the form of animation. Animation of changing content is used here to present the evolution of pages. It especially suits our passive viewing mode as it allows for a smooth transition between sequential page versions. It also draws users' attention to the updated content. The sequence of change presentation depends on the importance of changes in a page. We assume a simple estimation of change importance based on analyzing a

page from left to right and from top to down. A page is gradually processed so that deletions are shown first followed by additions. Content that is static between consecutive page versions remains displayed on the page throughout the transition. However, its location may change depending on the structure of the page in the latter page version. User can control the speed of the animation through a drag-and-drop meter. Negative velocity values are supported to enable browsing backward in time. After the current page transition is completed, the browser signals the completion of animation by a short sound and waits a moment presenting the complete page version. The length of this interval should be optimally chosen to enable the user to roughly grasp the contents of the currently displayed page version without significantly delaying the overall browsing process. It could be manually set by the user or alternatively could be made dependent on the type of the page and the amount as well as the distribution of its changes. Once the complete page version is presented, the browser analyses the next page version and again animates the changes. Thanks to this presentation method, we can demonstrate the evolution of page content in a smooth and intuitive way while at the same time allowing some time for the user to quickly skim through the page content.

We propose using a simple form of animation, that is, the effects of appearing and disappearing with varying blinking speed. For example, first a deleted element starts blinking slowly and then blinking speed increases until the element disappears. In addition, the insertion and deletion changes are indicated by different colors for their better perception and differentiation. Other animation effects could be utilized based on the user's preferences, time requirements, etc.

As mentioned above, there is a manual time jumping capability provided by enabling the user to input a new date or to click on any point in timeline. However, because of the abundance of static, redundant content in past Web pages, we also propose an automatic jumping facility. This is automatically adapted to the page contents and to the distribution of changes in time. During the presentation flow, the browser jumps over the changeless periods and displays the first page containing any content change (Figure 2). Automatic jumping facility can be switched on or off by the user. By equipping the browser with this function, the browsing experience is improved thanks to limiting the displayed page versions to the ones that contain changes.

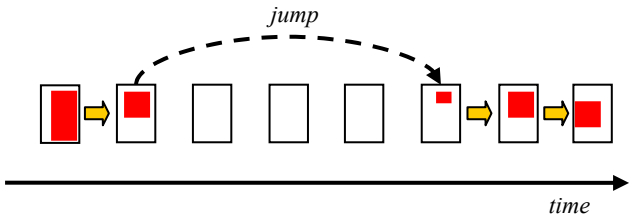

Figure 2 Automatic jump (horizontal browsing is represented by block arrows; changes in page versions are shown as small rectangles)

3.4 Vertical Navigation

The user can stop the presentation by pressing either the pause or stop button. The former halts the animation and displays a partially rendered page. The latter interrupts the presentation, but orders the browser to load the consecutive page version, without completing the change presentation.

While the animation is interrupted, the user can read a page in detail and optionally follow the links on the page. When a link is clicked the browser queries archives to find the available version of the target page that is closest in time to the actual time point of browsing. Then, when the user presses the play button, the browser starts presenting the new page. If the new page is not stored in the archive, the browser signals an error by a short sound or visual signal.

The back button is one of the most often-used navigation tools in browsers for the live Web. It is based on the stack mechanism for the browsing history. Two back buttons are prescribed in the framework for enhancing vertical navigation: a page-version-consistent button and a time-consistent button. The paths traversed by using each of the back buttons are compared in Figure 3. The first button returns the display to the page version from which the link was clicked, in much the same way as with browsers for the live Web. However, in the past Web it often means also jumping backwards or forwards in time depending on the direction of browsing. The second button returns the display to the version of the previously viewed page that is closest in time to the version from where the user started clicking the back button. This type of back navigation thus moves in a vertical way and minimizes time variance between visited pages.

Additionally, we also provide two forward buttons working in the similar style as the back button.

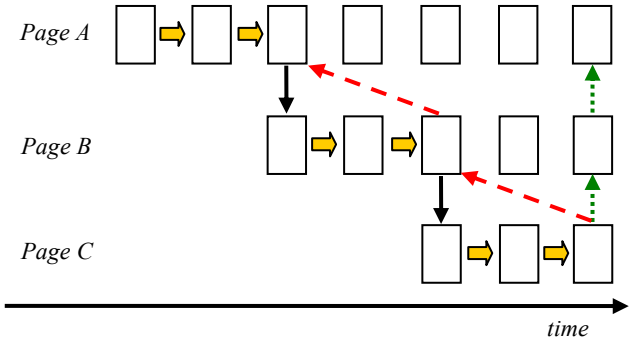

Figure 3 Vertical navigation by using two back buttons (dashed lines represent page-version-consistent back navigation and dotted lines represent time-consistent back navigation; block arrows show horizontal navigation and solid lines represent vertical navigation)

4. SEARCH-ORIENTED BROWSING

In this section we propose filtered browsing by using query-based and localized filtering. These enhancements are designed for information-seeking users to minimize the time spent searching for information by automatically filtering and limiting the information displayed.

4.1 Query-based Browsing

By choosing query words the user indicates the kind of changes in which he or she is interested. The system can then outline those textual changes that are highly similar to the query. The simplest and most cost-effective implementation is animating particular changes only if they contain any of the words found in the query. The changes, which do not contain query words, would be treated as static text and displayed immediately without any animation. Additionally, the automatic jumping mechanism could be used to

view pages with only query-related changes. This kind of filtered browsing should enable users to find changed content of interest and guide them through the past Web.

4.2 Localized Browsing

Many pages feature "topical blocks" that display content topically related. The main pages of news sites are a good example of such a structure. The news stories are often divided into well-understood, intuitive categories, such as, for example sport, business or politics. Localized browsing concept is based on this observation and provides a mechanism for selecting the area of the page for horizontal browsing. For example, one can choose the sports section on a news page. The browser then displays a new window containing only the desired section and starts the horizontal browsing. In this way, only changes appearing in the selected area of the page are shown. Automatic jumping mechanism together with other browsing facilities can be used in the new window. The browser is thus equipped with a spatial filtering mechanism. The user could also combine localized browsing with query-driven browsing for even more precise content filtering.

The localized browsing mechanism uses information about the layout of the selected content in order to find the minimal HTML block that encompasses the selected area. If the user selects an area spanning more than one HTML block, the system considers the minimal part of the page structure that embraces the entire user-selected area. With this sort of browsing, the user predicts the type of content appearing in the specified page area based on the assumption that the layout of the browsed page does not change. Localized browsing will fail if there is a considerable change in the page structure, and the user should be made aware of this potential danger. One way to overcome this problem could be the analysis of content similarities for tracking the topical blocks.

5. SYSTEM IMPLEMENTATION

The browser was built in C# using Microsoft Internet Explorer Component; its interface is shown in Figure 4. Since this is only a preliminary test version, we have focused more on functionality of the browser and less on its usability. In the future we plan to improve the user interface design.

We start the description of the implementation of the browser from the change detection mechanism. Each selected past page version was compared with its next version of the page for detection of any changes. First, both page versions had their HTML sources reformed so that each HTML node was converted into one text line. We used the diff algorithm for change detection [10]; it is based on longest common subsequence detection. We decided to utilize the diff algorithm, as it is a popular method for computing difference between documents and can be easily implemented. Its drawback is that it detects any kinds of changes in the text including also typos. However, generally, it is difficult to predict the types of changes and their degrees that can be interesting to different users for different pages. Nevertheless, more complex algorithms for change detection could be utilized. Image changes were detected by comparing *src* and *alt* attributes in image tags. For marking changes, we used inline HTML tags, <ins> and <del>; they were inserted based on HTML grammar rules.

The change animation was realized by switching interchangeably style settings of visibility of HTML nodes. First, the system selected the first line containing a node with <ins> or <del> tags. In case of <del> node the system set its visibility to "hidden". Short time later the visibility of the node was set to "visible". After repeating this process 10 times, the system changed the style of the node to "display: none" in order to finally hide the node and to reuse the space that was occupied by it. Then, all the page content that has not been rendered yet was shifted upwards to fill the freed space. The time period between consecutive changes of the node's style settings was decreasing gradually from 300 to 50 milliseconds to provide the effect of the increase of blinking speed. <ins> node was animated in a similar way but in reverse order. First, the below content of the page was shifted downwards to make the space for the new insertion. Then, the style setting of visibility of the node was switched interchangeably 10 times with decreasing speed. Finally the display of this node was changed to "inline". Besides the animation effect, the added and deleted changes were also indicated by different background colors. In this way, the system processed gradually all following lines containing <ins> and <del> tags to complete the animation.

To enable query-based browsing the similarity between a query and change text was computed by identifying words they had in common. Due to efficiency concerns, we did not decide to implement stemming and stop-word filtering. Localized browsing was not implemented in the prototype system.

It sometimes took a long time to browse all changes one by one, so we equipped the browser with a "change all at once" option to enable faster browsing by visualizing all changes at once. Also, for pages with a high number of versions, it was difficult for users to use the timeline and to see the version dates. We thus added a scrollable list of all available page versions (see Figure 4) with information about their dates. The list functionality is similar to that of the timeline, enabling a user to click on and to move to a target page version.

Since the content of pages that was larger than the browser's window could not be viewed at once, we had to modify the presentation style by adding another option. The user could either anchor the viewpoint of the page or scroll the page automatically. With the first choice, only the visible part of the page can be browsed; with the second, the animation is presented for the entire page before the next page version is displayed. Therefore, with the first, the user might miss important information residing on the unseen part of the page, although the overall browsing is faster. With the second, no content is skipped, but the presentation might take longer time.

As the speed is an important concern for a browsing application for the past Web, we advocate an "aggressive" prefetching and caching policy to realize smooth browsing. Prefetching and caching are widely used mechanisms for improving browsing in the live Web [8,9]. During horizontal browsing the system prefetched up to some number of next page versions so that they could be later retrieved from cache during subsequent steps. Additionally, when the user pressed the pause or stop button, the browser analyzed the current page version and triggered the link prefetching process. Link prefetching is used by several Web browsers such as, for example, Mozilla [21]. This utilizes the idle time of the browser for downloading pages that have a high probability of being visited by the user. Our framework prescribes prefetching of only the lists of available versions of the pages that have links posted on the currently viewed document, however up

to certain number of page versions could be prefetched. The sequence of links for prefetching was based on their location on the page. The ones at the top-left were prefetched first. For query-based browsing link prefetching was limited to those links with the content in anchor areas that contained the query words. The prefetching method used in the current implementation is rather simplistic; more complex algorithms could be proposed, such as ones based on page content, a user model, or the browsing history.

6. EVALUATION

We evaluated the validity of our framework experimentally using a prototype past Web browser. We used the Internet Archive as the main repository of past page versions. Additionally, previous page snapshots were collected from Google and MSN Search caches, as well as from a local cache. Two pilot experiments were carried out on a personal computer with a Pentium M 1.7-GHz CPU and 2.0-GB RAM to test the functionality of the browser.

6.1 Experiment 1

In the first experiment, we compared our browser with the Wayback Machine Web-based interface. Eight participants were asked to find specific pieces of information on certain Web pages. They were divided into two groups: the groups alternated between using the Wayback Machine and the past Web browser to perform four tasks. Each subject performed an equal number of tasks using each interface. They did not have any experience using either interface. They were thus given an explanation of each and 10 minutes of training prior to task execution. Each task lasted a maximum of 10 minutes, except for task 3, which lasted a maximum of 5 minutes. Below we briefly describe the tasks.

Task 1 and ***Task 2***: The first two tasks were aimed at examining the usefulness of the passive browsing concept together with the change detection and presentation. In the first task, the subjects browsed horizontally five versions of the Yahoo! main page taken from the period May 28–June 6, 2000. They were told to find all the changes and answer three questions related to them. The pages were rich in content but with relatively few changes during the target period. In the second task, they horizontally browsed nine versions of the Open Directory Project main page taken from June 22–October 17, 2000. These pages had little content and relatively few changes. The subjects were told to find and describe all the changes and identify the part of the page that changed most during the target period.

Task 3: The aim of the third task was to check the effectiveness of query-driven browsing. The subjects were told to find changes in a Web page using the query-driven approach with the past Web browser or using the Wayback Machine. The Web page was the homepage of a University of Tokyo laboratory. They browsed 34 page versions for the period from January 19, 1997, to October 13, 2004. The pages were similar to those in the first task—rich in relatively static content. The task consisted of finding changes related to the names of two laboratory members. For simplicity, the subjects were told to focus on the laboratory-members section of the page.

Task 4: The aim of the fourth task was to compare the effort needed during browsing of the past Web vertically and horizontally with both systems. The subjects were told to follow a path through the past Web: browse the Yahoo! main page horizontally, follow a certain link to another page, browse that page horizontally, and then follow another link again and browse the third page horizontally. The time period covered was August 6, 2000, to November 11, 2002. The final step was to return to the starting page via the page versions they had browsed vertically.

We counted the number and calculated the percentage of correct answers for each task. Not all tasks were completed with the Wayback Machine as some subjects found them too difficult and had quit before finishing them (task 1, three subjects; task 2, two subjects). We also measured the time and effort expended by each user for each task. Tables 1 and 2 show the average results for the Wayback Machine and the past Web browser, respectively. The results were significantly better for the past Web browser for each task. On average, it took less time, fewer keystrokes, fewer mouse clicks, and less mouse movement to complete the tasks with the proposed browser. The advantage of the passive style of the proposed browser can be seen in the statistics on user effort (number of keystrokes, number of mouse clicks, and amount of mouse movement). Moreover, the shorter times taken, together with the higher percentage of completion for the first three tasks, indicate that the past Web browser is an efficient tool for finding specific information about changes in past Web pages with varying degrees of content richness.

The results for task 3 show that the query-driven approach is an effective way to browse relatively large number of content-rich past page versions. With the past Web browser, the task was completed very quickly with minimal user effort considering the large number of page versions. The results for task 4 demonstrate that our navigation tools enable faster and easier travel in the spatio-temporal structure of the past Web. With the Wayback Machine, the subjects always had to return to the main directory page, which lists the available page versions, when they wanted to browse the past Web horizontally. Vertical navigation was even more troublesome.

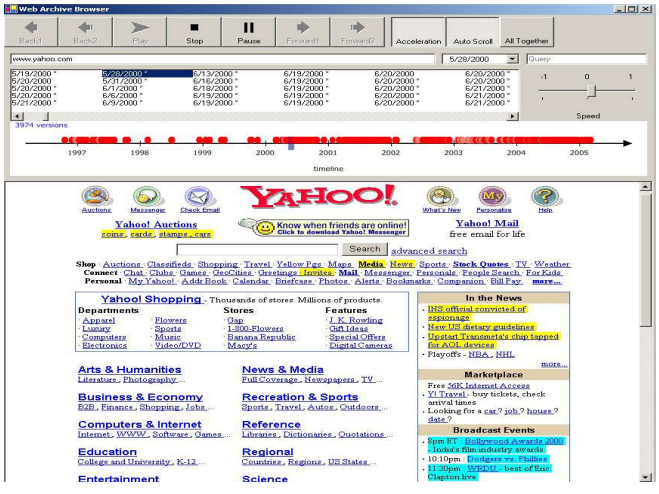

Figure 4 Interface of past Web browser

Table 1 Results for the Wayback Machine

Task	Completion rate (%)	Time (min:sec)	Keystrokes	Mouse clicks	Total mouse movement (m)
1	45	7:54	13.5	22.5	9.45
2	28	5:0	0	87.3	14.04
3	48	9:25	2.75	62.5	14.1
4	55	8:5	66.5	54	16.37

Table 2 Results for past Web browser

Task	Completion rate (%)	Time (min:sec)	Keystrokes	Mouse clicks	Total mouse movement (m)
1	100	5:39	0	0.25	0.05
2	85	2:26	0	2	0.61
3	100	1:45	0.75	3	0.57
4	93	2:40	0	12.5	2.85

6.2 Experiment 2

In the second experiment, we asked the subjects to freely browse past versions of their favorite Web pages during 10-minute sessions. Then the subjects were asked to answer three questions and provide comments.

1. Was it easy to understand how to use the past Web browser?
2. Was it easy to use the past Web browser?
3. Would you prefer to use the past Web browser over the Wayback Machine for browsing the past Web?

The results are shown in Figure 5. The results for the first question show that the subjects generally agreed that it was quite easy to understand how to use the proposed browser. The majority of subjects attributed this to the fact that the browser interface is quite similar to traditional browser and video player interfaces.

However, the results for the second question demonstrated that, in practice, the proposed browser was not so easy to use. The many options and buttons made it troublesome for users to browse the past Web efficiently. Some complained that too many page versions were indicated on the timeline, which made it difficult to visualize the distribution of pages and their changes over time. Also, they would have preferred colors sharper than the red and pink that were used. Some said that better indication of what the browser was doing would help, especially during longer processing periods. One user reported problems with understanding the functioning of the two back buttons. We believe that making the necessary adjustments to the system and simplifying the GUI together with a better explanation and more browsing experience would increase user ability to browse the past Web using the proposed browser.

The last question concerning user preference indicates that, on average, they preferred past Web browser to the Wayback Machine. According to the subjects, they preferred past Web browser mainly because of the change detection and presentation by animation, which gave them an intuitive feeling for the page evolution. They also liked the shorter browsing time and less effort needed to traverse the past Web.

Incidentally, we observed that some users expressed a feeling of nostalgia when browsing past versions of their favorite pages. Browsing the past Web is evidently attractive.

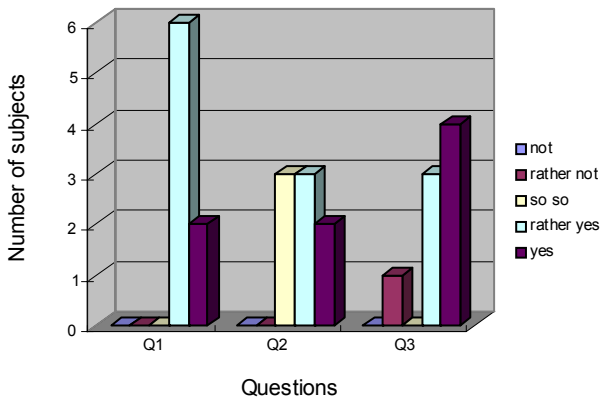

Figure 5 Questionnaire results

7. DISCUSSION

Our framework proposal has several limitations due to the assumptions and simplifications that were necessary. Below we discuss them.

- Assumption that archives can deliver available page versions with their timestamps upon a request containing a time period and page URL. We expect there will be more past Web repositories available in future enabling public access to stored versions of pages.
- The result of the history reconstruction of a page depends on the amount and the quality of its past snapshots stored in Web archives. If the page has poor representation of its past states among the available repositories then the browsing may not be satisfactory. Besides that, dynamic applications using such technologies like, for example Flash or JavaScript may not be working properly. We also assume permanent URL addresses of pages over time. URL address can sometimes change during page lifetime even if the page remains basically the same. To cope with this problem, we need to provide a method for detecting changes in the URL addresses of pages.
- Pages may have few or no changes at all, so viewing them horizontally can be boring. This problem is evident, for example, in news sites where, often, different articles are created as separate Web pages that can be accessed from main pages. An opposite problem occurs when a page has too many changes, and it takes a long time to browse the page in time. Automatic jumping and filtered browsing can help to alleviate these problems.
- The proposed change detection mechanism is not perfect and should be improved. In addition, the browser may not work properly for web pages with relatively large changes of layout over time.
- Efficiency of browsing can be hindered by poor network resources. Prefetching should be used to alleviate this problem.

Nevertheless, this framework makes several important contributions.

- It is a client-side system that incorporates a meta-archive approach to increase the overall coverage of data in Web archives resulting in a unified access to the past Web.
- It provides a means for observing the evolution of pages as well as a means for easy, customized browsing of the past Web. This is realized by:
    - Interface similar to video players with passive viewing of page transitions over time.
    - Browsing based on change detection and presentation in the form of animation to allow a smooth transition between page versions and to draw user's attention to updated content.
    - Navigation tools such as timeline, back and forward buttons, and automatic jumping mechanism enabling the user to skip periods without change.
- It supports filtered navigation thanks to:
    - Selective browsing through query-driven filtering.
    - Localized browsing for limited viewing of a user-selected area.

One potential application of the proposed browser assumes integration of the past and live Web for browsing at the same time. The concept of combining browsing of archived data with browsing of the live Web has been actually implemented in WAXToolbar - a Firefox browser extension to the Wayback Machine [29]. If the interface to the live Web is added to our proposed framework then users will be able to browse current versions of Web pages in the traditional way as they are doing now. Present versions of pages would be inserted into the timeline and accessible as any other past version. When necessary, users could see the histories of interesting pages by browsing backward in time. In this way they would have potential starting points for their journey to the past. Below we list some possible applications of the mixed-Web browsing using our proposed framework:

- Watching the history of an interesting page or its part in order to see its past content. Users may want to view more content from pages that are particularly attractive to them. They might be searching for certain content that already disappeared from a page. For example, one remembers that he or she had seen a particular article on the page and wants to read it now in detail, however, it is no longer available in the live Web. In this case the user can travel back in time to try to find the article by using query-driven browsing. In addition, one can also see the past context of an element from the present version of the page. To better understand a particular part found on the present page the user may travel back in time to the point of the element's origin in order to see its local context at that time.
- Retrieving pages that cannot be accessed. This case is similar to Google's cache system. Many times users use the cache in the search engine to view the page that is currently inaccessible. Such usage can be realized for any page provided that its recent version is stored in any available online Web archive.
- Detecting age of page elements. For example, upon noticing a person's name in a laboratory members' list in the present version of a page, the user may want to know when the particular person was first listed in this page. However, in many cases such information cannot be easily obtained. Thus, he or she can use query-based browsing to approximately discover when the person started working in the particular laboratory. In another example, a user may wish to know the date of the occurrence of a statement in the news section of a company Web page to check if it is actually new. In short, each element of the page content can be associated with its approximated date of origin provided that the history of the page can be reconstructed with a sufficient precision. This approach provides temporal information that can shed new light on the page content.

8. CONCLUSIONS

In this paper we have described a general framework for a past Web browsing interface that supports horizontal and vertical browsing. We propose a client-side browser that enables easy and customized viewing of the history of pages. The framework incorporates the meta-archive approach for improving the coverage of Web archives. It requires real-time communication with complementary past Web repositories. The framework features passive browsing style and utilizes change detection and change presentation. Since the changes in content are of particular importance when browsing past pages, the browsing is based on emphasizing the changed content. The changes are animated to smooth the transitions between page versions and to draw the user's attention to updated content in the page over time. The user can control the presentation flow in a way similar to video players. The framework also includes search-oriented functions for improved browsing by filtering information. They enable the user to specify an interesting change or an area to be watched. Finally, we discuss the potential applications of combined browsing of the live and past Web.

In future we plan to work on overcoming the limitations mentioned in the paper. We also hope that our efforts spark more interest in the Internet community in developing more efficient interfaces for browsing and searching the massive amounts of past data stored in Web archives.

9. ACKNOWLEDGEMENTS

This research was partially supported by the Japanese Ministry of Education, Culture, Science and Technology Grant-in-Aid for Scientific Research in Priority Areas entitled: Content Fusion and Seamless Search for Information Explosion (#18049041, Representative Katsumi Tanaka), and by the Informatics Research Center for Development of Knowledge Society Infrastructure (COE program by the Japanese Ministry of Education, Culture, Sports, Science and Technology) as well as by the Japanese Ministry of Education, Culture, Science and Technology Grant-in-Aid for Young Scientists B (#18700111).

10. REFERENCES

[1] AOLA. Austrian On-Line Archive: http://www.ifs.tuwien.ac.at/~aola

[2] Arvidson, A., Persson, K. and Mannerheim J. The Kulturarw3 project - The Royal Swedish Web Archiw3e - An example of "complete" collection of Web pages. In

Proceedings of the 66th IFLA Council and General Conference, Jerusalem, Israel, 2000, 13-18.

[3] Boyapati, V., Chevrier, K., Finkel, A., Glance, N., Pierce, T., Stockton, R., and Whitmer, C. ChangeDetector™: A site level monitoring tool for WWW. In *Proceedings of the 11th International World Wide Web Conference*. Honolulu, Hawaii, USA, 2002, 570-579.

[4] Brewington, B. E. and Cybenko, G. How dynamic is the Web? In *Proceedings of the 9th International World Wide Web Conference*. Amsterdam, The Netherlands, 2000, 257-276.

[5] Chi, E. H., Pitkow, J., Mackinlay, J., Pirolli, P., Gossweiler, R. and Card, S. K. Visualizing the evolution of Web ecologies. In *Proceedings of Conference on Human Factors in Computing Systems (CHI '98)*, Los Angeles, California, USA, ACM Press, 1998, 400-407.

[6] Cho, J., and Garcia-Molina, H. The evolution of the Web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB)*, Cairo, Egypt 2000, 200-209.

[7] Cockburn, A. and McKenzie, B. What do Web users do? An Empirical Analysis of Web Use. *International Journal of Human-Computer Studies* 54(6), Academic Press, Inc. 2001, 903-922.

[8] Davison, B. D. A Web caching primer. *IEEE Internet Computing*, 5(4), 2001, 38-45.

[9] Davison, B. D. Predicting Web actions from HTML content. In *Proceedings of the 13th ACM Conference on Hypertext and Hypermedia*, College Park, MD, USA, 2002, 159-168.

[10] Diff Algorithm: http://www.codeproject.com/cs/algorithms/diffengine.asp

[11] Douglis, F., Ball, T., Chen, Y., and Koutsofios, B. AT&T Internet difference engine: tracking and viewing changes on the Web. *World Wide Web Journal*, 1(1). Kluwer Academic Publishers, 1998, 27-44.

[12] Dyreson, C. E. Towards a temporal World Wide Web: a transaction-time Web server. In *Proceedings of the 12th Australian Database Conference (ADC '01)*, Gold Coast, Australia, 2001, 169-175.

[13] Dyreson C. E., Lin H-L, and Wang Y. Managing versions of Web documents in a transaction-time Web server. In *Proceedings of the 13th International World Wide Web Conference*, New York, USA, 2004, 422-432.

[14] Election Web 2002 Archive: http://lcWeb4.loc.gov/elect2002

[15] Fetterly, D., Manasse, M., Najork, M., and Wiener, J. L. A large-scale study of the evolution of Web pages. In *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary, 2003, 669-678.

[16] Grandi, F. *An Annotated Bibliography on Temporal and Evolution Aspects in the World Wide Web*. TimeCenter Technique Report, 2003.

[17] Hallgrimsson, Þ. and Bang S. "Nordic Web Archive" In *Proceedings of the 3rd ECDL Workshop on Web Archives* in conjunction with *the 7th European Conference on Research and Advanced Technologies in Digital Archives*, Trondheim, Norway, 2003.

[18] International Internet Preservation Consortium: http://netpreserve.org

[19] Internet Archive: http://www.archive.org

[20] Jacob, J., Sanka, A., Pandrangi, N. and Chakravarthy, S. *WebVigiL: An Approach to Just-in-time Information Propagation in Large Network-centric Environments*. In Web Dynamics Book. Springer-Verlag, 2003.

[21] Mozilla Web Browser, Link Prefetching FAQ: http://www.mozilla.org/projects/netlib/Link_Prefetching_FAQ.html

[22] NWA Toolset: http://sourceforge.net/projects/nwatoolset http://nwa.nb.no

[23] Offline browsing: http://en.wikipedia.org/wiki/Offline_browser

[24] Pandora, Australia's Web Archive: http://pandora.nla.gov.au

[25] September 11 Web Archive: http://september11.archive.org

[26] Tauscher, L. and Greenberg, S. How people revisit Web pages: empirical findings and implications for the design of history systems. *International Journal of Human Computer Studies*, 47(1), 1997, 97-137.

[27] Toyoda, M., Kitsuregawa, M. A system for visualizing and analyzing the evolution of the Web with a time series of graphs. In *Proceedings of the 16th ACM Conference on Hypertext and Hypermedia*, Saltzburg, Austria, 2005, 151-160.

[28] Toyoda, M., Kitsuregawa, M. Extracting evolution of Web communities from a series of Web archives. In *Proceedings the 14th ACM Conference on Hypertext and Hypermedia*, Nottingham, UK, 2003, 28-37.

[29] WAXToolbar: http://archiveaccess.sourceforge.net/projects/waxtoolbar

[30] Wayback Machine Open Source Implementation: http://archive-access.sourceforge.net/projects/wayback

[31] WERA: http://archive-access.sourceforge.net/projects/wera