Collaborative Document Monitoring

Natalie Glance [♣♥] Jean-Luc Meunier [♠] Pierre Bernard [♠] Damián Arregui [♠]

[♣] WhizBang! Labs - Research
4616 Henry Street, Pittsburgh
Pa 15213, U.S.
nglance@whizbang.com

[♠] Xerox Research Centre Europe
6 chemin de Maupertuis
38240 Meylan, France
{firstname.lastname}@xrce.xerox.com

ABSTRACT
In this paper we present a second generation URL monitoring tool which enables the collaborative evaluation of URL content changes. In our implementation, a document monitoring agent works alongside a recommender system. Using information provided by the monitoring agent, the collaborative system alerts users when documents they are monitoring have changed. The monitoring agent provides automatic evaluation of the nature of the change. Users, however, add subjective evaluations; one user's effort informs all others monitoring the same URL. Based on these subjective evaluations, the collaborative system can filter the changed URLs, providing customized notifications for each user based on individual preferences. In this paper, we describe the implemented system and usage results.

General Terms
Design, Experimentation, Human Factors.

Keywords
URL monitoring agent, recommender system, WWW

1. INTRODUCTION
Web monitoring tools [1][4][6][7][15][18][19] have played an important role since the early days of the WWW. By automatically tracking changes in URLs, these tools allow users of the Web to more easily keep up with dynamic content. With the advent of the mobile Web, monitoring tools have found another niche: notification via mobile phone of changes in stock prices, news about specific companies or markets, availability of a certain item at an Internet-based auction, and much more.

Monitoring of structured information sites, in which the Web page is the front end for a server-side database, is relatively straightforward. Changes identified by the monitoring agent are likely to be significant and have semantics associated with them. Monitoring of unstructured or semi-structured sites is more problematic, however. Despite advances in techniques for detecting structure in pages [13], errors obvious to the human eye will continue to occur. Thus, the monitoring agent may be fooled by layout changes, or banner changes, while not able to detect seemingly minor word changes as important.

Examining change histories of pages in directories such as that maintained by *DailyDiffs* [4] (see, for example, Fig. 3b) makes it clear that meaningful content changes represent only a small fraction of all detected changes. The larger proportion of changes are insignificant, such as a "last modified" date change, or a new ad banner, or perhaps a change in font size or layout. Thus, the results returned by a URL monitoring agent risk having a small signal to noise ratio.

Furthermore, what defines a change as meaningful is subjective. Two people, both monitoring an on-line computer catalog, may have very different evaluations of changes to the same page: one may be waiting for the price on a component to drop, the second may be waiting for another component to be marked "in stock." This example also shows that it would be wrong to assume that a small change is necessarily unimportant.

In addition, with large numbers of WWW information sources now available, people require a way to summarize new available information. One way is to depend on editorial services. A second way is work together with others interested in monitoring the same sources for the same kind of information. For many sources of information, a URL of interest may have thousands, if not millions, of readers. A URL monitoring tool that manages to harness the collaborative nature of the Web, can help groups of people monitor URLs together for significant changes.

In this paper, we present the collaborative monitoring system which we have developed on top of an existing recommender system. The underlying recommender system allows user to share and recommend URLs of interest to communities of interest. We have implemented a document monitoring agent that works alongside the recommender system to alert users when recommended documents they are monitoring have changed.

The monitoring agent provides an automatic evaluation of the nature of the change. This automatic evaluation guides user evaluation; it indicates, for example, what percent of the content has changed, how many links have been added/removed, what percentage of the keywords have changed. Users, however, provide the subjective evaluation and one user's effort informs the

GROUP'01, Sept. 30-Oct. 3, 2001, Boulder, Colorado, USA.

[♥] Work performed while the author was at Xerox Research Centre Europe.

whole community: "It's time to buy - the price has dropped 20%!" or "The authors have added a new section on¼"; or "The only thing that's changed is the layout." Based on these subjective evaluations, the recommender agent can decide which changed URLs to report to each user and which to set aside.

In the next section, we first describe the current state-of-the-art in monitoring systems and present our in-house document monitoring agent. In Section 3, we discuss collaborative document monitoring and stress the various user interface and process issues we encountered designing the system. In Section 4, usage statistics from our system our presented. Finally, we conclude by discussing future directions.

2. DOCUMENT MONITORING

2.1 History

The first Web monitoring tools appeared in the early 1990s, allowing users to register URLs and receive notifications when the contents had changed. Given how dynamic web pages can be, this kind of service is extremely useful for keeping up-to-date. Examples of URLs that are useful to monitor include: product listings, e-zine indices, search results for a given query, software release histories, R&D project homepages or publication lists, insider trade listings, company press release pages, Web site "What's new?" pages, friends' homepages, "Website coming soon…" pages, company job listings, and more. Some of these sites may provide e-mail notification of changes. However, using a monitoring tool allows users to centralize all alerts into at most one e-mail notification per day, and also potentially allows users to maintain their privacy with respect to the different sites they monitor.

URL-minder [15], now called Mind-it, was one of the first such tools. It offered its monitoring service for free on the Internet, and within a few months already had thousands of users. It worked as follows: users registered their URL of interest on the URL-minder Web site. In turn, the server fetched the pages on a regular basis to detect changes and notified the user by e-mail. Current monitoring tools work similarly and can be categorized into server-based or client-based tools.

Mind-it belongs to the first category; the tool runs on a server machine supporting multiple users. Server-based tools have the advantage of accessing only once multiply registered URLs, e.g., popular Web pages. On the other hand, privacy becomes an issue: users may opt not to register certain URLs because they are reluctant to reveal their pages of interest to a third party. Pages with restricted access also cannot be handled server-side while maintaining user privacy; this is the case for pages requiring a login/password or cookie-based identification.

More recently, server-side tools have been packaged to provide added value on e-commerce sites. NetMind, the company behind Mind-it, has a number of customers including eBay, RedHerring and Siemens, among others. The auction alerts provided by eBay are powered by NetMind's monitoring technology. In addition, with the advent of mobile e-commerce, delivery options for change notifications have been extended to mobile platforms.

Client-based monitoring tools, on the other hand, run on the user's own machine, and therefore preserve privacy and properly manage restricted-access pages, but globally consume more network resources. Network load could become an issue for organizations whose members do client-based monitoring, but could potentially be reduced if the sets of pages being monitored overlap significantly. Client-based tools can also pre-fetch modified pages and related resources (images, sounds, and related pages), allowing users to browse pages off-line and/or with lower latency. Examples of client-side tools include SurfBot [18], Microsoft Internet Explorer and WebSpector [23].

Recent research work on monitoring tools [1][7][6] introduced novel architectures and features. First, WebTracker [7] allows users to share tracked URLs, while also allowing users to track private URLs. It uses the Unix *diff* tools to show differences to users. The Do-I-Care [1] agent employs relevance feedback. User feedback on their interest in the change detected by the agent is used to train the agent and improve its future performance. A Bayesian classification process is used to build a profile of the user interest and to score detected changes, which in turn are compared against a configurable threshold. Do-I-Care also proposes a novel architecture which allows agents to cascade among themselves the most relevant information, leveraging thus the work and judgment of other users. Collaboration among users is achieved via their agents and is possible only once users have trained their personal agents. The AT&T Internet Difference Engine [6] incorporated sophisticated methods for performing the difference analysis, using an adaptation of Hirschberg's solution for finding the longest common subsequence [11] to HTML pages. Finally, Pluxy [5] proposes a novel architecture: Pluxy is a Web proxy that hosts a dynamically extensible set of services; the authors propose, among others, a Web monitoring service.

2.2 Document Monitoring for Group of Users

Implementing a basic monitoring agent is fairly straightforward. We built our own, taking into account features required to enable collaborative monitoring and experimenting with in-house linguistic tools to do more sophisticated characterizations of content changes. We also wanted the monitoring agent to provide additional services through an API, allowing it to connect to other modules, in particular to a collaborative system for sharing URLs and other documents.

Thus, we required that our monitoring agent to be able to:

- *Synchronize the monitoring of pages*: reference versions are saved each day, the first of every week and the first of every month; as a result a group of users can monitor the same page with the same frequency (daily, weekly, monthly) and see the same set of changes. Users who monitor the same page with different frequencies will not see the same changes. An API is provided for notifying other modules of the extent of the change as characterized by percentage changes in content, text, noun phrases, links and images (described further below).
- *Identify broken links*: the monitoring agent reports broken links and detects URLs that have moved. To account for possible transient network failure, a counter of failed download attempts is associated to each registered document. A configurable threshold determines when to alert other modules that the document has disappeared with high probability. URLs that have moved (and that are reported as thus by the HTTP server) are dealt with in two different ways: for permanent moves, an API is provided for notifying

the other modules immediately, while temporary moves are dealt with silently.

- *Identify duplicates*: the monitoring agent actively searches for cases in which the same document is registered twice with different URLs. It provides an API for notifying other modules when it detects duplicates.
- *Handle multiple clients*: the monitoring agent provides these services on a per client basis, where a client can be a single user accessing the monitoring agent through a simple HTML interface, or a software module communicating with the monitoring agent via HTTP.

The handling of broken links and duplicates allows other modules to keep current their store of URLs. How the broken links and duplicates are dealt with by other modules is left to their discretion. An example policy, implemented in our system, is discussed in the next section.

Our second reason for implementing our own monitoring agent was to provide a set of user-customizable policies for detecting changes. In particular, we wanted to test how well linguistic analysis of content changes could perform. Clients of the monitoring agent can request the agent to monitor any (or all) of the following types of content change:

- *Any change*: The monitoring agent compares documents at the byte level. This is the lowest level detection.
- *Text change*: HTML tags are removed, and then an inverted index is constructed for the document. The percent change in the inverted index is reported. Thus, changes in the document layout and presentation are ignored.
- *Noun phrase change*, *token change*: after removing HTML tags, noun-phrases and tokens are extracted by using Xelda [20], XRCE's platform of linguistic services (such as noun phrase extraction and language guesser). The number of occurrences of the most frequent noun-phrases (tokens) in the two versions of the page is then compared using the following formula, whose results range from 0 (no difference) to 1 (completely different):

$$diff = \frac{\sum |n^2 - n'^2|}{\sum (n+n')^2}$$

  with n and n' being respectively the old and new occurrence of a noun phrase (token). By setting a threshold, it is then possible to tune the agent's sensitivity.
- *Link change*, *image change*: The agent monitors changes to hypertext links and images for an HTML document.

Currently, these change-detection policies have been implemented only for HTML documents and not for postscript, pdf, etc.

The monitoring agent maintains the internal representation of each registered document (size in bytes or inverted index) which is appropriate to its associated detection policies (different policies may be selected by different users for the same document). The agent also caches the latest version of each document, without images.

3. COLLABORATIVE MONITORING

The important novelty of our work is making document monitoring a collaborative activity. In order to implement collaborative monitoring, we integrated the monitoring agent with a workplace recommender system called Knowledge Pump [8][9], which helps users share recommendations of URLs. Knowledge Pump (KP) allows users to submit recommendations of URLs, local files (via upload), or text. A recommendation consists of a rating and, optionally, a comment. In turn, KP calculates personalized sets of recommendations for all users, based on similarities in ratings profiles. It is a natural extension of KP to allow users to monitor URLs they recommend or have had recommended to them. The addition of a document monitoring agent allows the recommender system to handle the entire lifecycle of document recommendation, from initial recommendation, to collaborative evaluation of changes to a document, to notification of changes to a document and of new reviews.

3.1 Process and UI: Design Issues

In order to integrate the monitoring agent with the recommender system, a number of questions had to be addressed, mostly focused around process and user interface:

- How frequently should documents be monitored? Who decides upon the frequency?
- What kinds of changes should be reported (any change, text change, image change, link change, etc.)? Should each user be able to select among the types of changes to report?
- Are all changes reported to users, or can the user set a threshold? Is the threshold fixed per URL or can each user choose? How should changes be reported to the user? Should we separate or combine user evaluations and agent evaluations?
- How should broken links and duplicate links be handled?

3.1.1 Monitoring Frequency

As stated previously, the document monitoring agent on its own allows clients to choose the monitoring frequency: either daily, weekly or monthly. Thus, in principle, different users could monitor the same URL in KP with different frequencies. However, if two or more people monitor the same URL with different frequencies, it becomes difficult for them to share evaluations of changes in the document. For example, say Paul and Claire are both monitoring the list of job openings at company X, but Paul requests daily alerts, while Claire requests weekly alerts. When a new job posting appears, Paul is likely to see it immediately and evaluate the change as important. This information will help Claire keep up-to-date on the page. However, by the time she evaluates the change, her comments will be stale from Paul's perspective and perhaps misleading ("another new job posting?" he might wonder). Thus, the sharing of evaluations works only in one direction when users monitor the same URL with different frequencies: the users monitoring with higher frequency help those monitoring less frequently; moreover, the comments from those monitoring less frequently may be misleading.

On the other hand, requiring users to monitor any given URL with the same frequency in order to avoid this kind of confusion raises different problems. First of all, who chooses the frequency? The first person to request monitoring of the URL? The person to request the highest frequency? Also, will people accept not

necessarily being able to choose the frequency of monitoring themselves?

In the first release of the system, we decided to enforce one frequency per URL, with the first person requesting to monitor the URL controlling the choice of frequency. That person also has the ability to later modify the frequency of monitoring, if, for example, s/he is unhappy with the initial choice, or if another user makes a request to modify the frequency. If the first user to monitor the URL later unsubscribes from monitoring the URL, the next person to subscribe or to change her/his monitoring settings for the URL inherits the power to modify the frequency.

However, user feedback from the first release strongly indicated that users prefer to individually choose the frequency of monitoring. Thus, for the second release of the system, we are implementing a process by which up to three different groups of users monitor a URL and share changes of evaluations among themselves, one group for each of the three frequencies. The change evaluations are propagated hierarchically: the weekly group is informed by a summary of the daily group's change evaluations and the monthly group is informed by a summary of the weekly group's change evaluations.

3.1.2 Monitoring Types

While the monitoring agent has the capability to monitor URLs for many different types of changes (any, text, noun phrase, image, links), including all of these possibilities within the framework of an existing system like KP might overwhelm the user. We decided to limit the types to: text, images and/or links. Our tests with users of the stand-alone monitoring agent client indicated that monitoring for text changes was more informative than monitoring for "any" change, while monitoring for "noun phrase" change tended to miss some significant changes that monitoring for "text" change caught. We found these subtleties too complex to make clear in a simple user interface within KP.

Again, the question arises of whether to allow different users to monitor for different kinds of changes or to require all users to monitor for the same kind. The issue is once again of ensuring that users can make sense of each other's change evaluations. The trade-off becomes even more difficult if the monitoring agent is extended to allow keyword monitoring or structured monitoring for changes in identified fields.

Again, we chose in our first release to favor the collaborative aspects over personalization. In the current implementation, the first user to request monitoring of a URL chooses the monitoring type for that URL. That user can later modify the monitoring type. If the first user to subscribe to a URL later unsubscribes, the next user to subscribe or to change her/his monitoring settings inherits the ability to modify the monitoring type. Interestingly, this policy did not receive the same negative response as did the policy requiring a single monitoring frequency. It is likely, however, that more sophisticated monitoring techniques would cause users to request personalization.

3.1.3 Change Thresholds and Change Notifications

The monitoring agent reports the percentage of change of any given kind (text, links, images). To provide rough semantics for the monitoring agent's calculation we mapped percentage ranges onto words: negligible means less than 5%; significant means between 5% and 20%; and important means greater than 20%.

Users are also able to qualify the extent of a change, but subjectively, via a rating and possibly a comment. Should they use a different vocabulary from the agent? We chose to have users employ the same vocabulary, characterizing a change as negligible, significant or important, but referring to nature of the change, not its size.

Should all changes be reported to the user? Should only changes above a certain threshold be reported? The problem with the latter is that even a small change detected by the monitoring agent may correspond to an important semantic change. Thus, we decided that users should be notified of all changes detected by the monitoring agent (pertaining to URLs that they have chosen to monitor). Once even one other user has evaluated the change, then this additional evaluation is used to filter the change notifications. Another user monitoring the same URL is notified only when the first user's evaluation is greater than his/her threshold.

The same vocabulary is employed by users when setting thresholds for monitoring URLs. Users can request to be alerted when changes are at least one of negligible, significant, important. If a user chooses, for example, to be notified of significant or greater changes, s/he will receive all change notifications generated by the agent alone, but only ones that are judged significant or greater by another user or group of users.

As there is no problem with regards to synchronization of threshold for notification among different users for the same URL, each user can choose his/her threshold individually.

3.1.4 Broken Links, Duplicate Links

There remains the question of how to handle stale URLs. Should fixing the problem be the administrator's role or the community's? On the one hand, giving the job to the administrator assures the security of the system; on the other, users may be better informed regarding the status of the URL and better able to update it. Within a work setting, it may make sense to distribute the work by permitting users to modify the URLs. Thus, in our current implementation, the user who first submitted the URL is notified if the link is determined to be broken or to be a duplicate. The user can then either modify the URL or request the system to retrieve the previous version from the document monitoring agent's cache and save it locally. Future requests to the stale URL are then redirected to the locally saved version. Monitoring can be turned off if it is believed that the URL has disappeared. The user can request that duplicate URLs be merged. In this case, the two groups of subscribers are merged as well.

3.2 Process and UI: Implementation

In integrating the monitoring agent with KP specifically, the two processes of collaborative monitoring and recommending needed to be merged in a way that would be transparent to the user. We modified the recommending/reviewing input to KP to add the possibility of monitoring. Figure 1 shows the review/recommend window from KP with monitoring added.

When recommending or reviewing an item, users can request to have that item monitored, if it is a URL. The default is that monitoring is unchecked.

The first user to monitor an item can select to monitor for text, link, and image changes on a daily, weekly, or monthly basis. Subsequent users choosing to monitor the same item will not be able to modify these settings; the UI displays a non-form based

summary of these monitoring settings. Each user monitoring the URL can choose his/her own threshold for notification: one of *negligible*, *significant* or *important* changes.

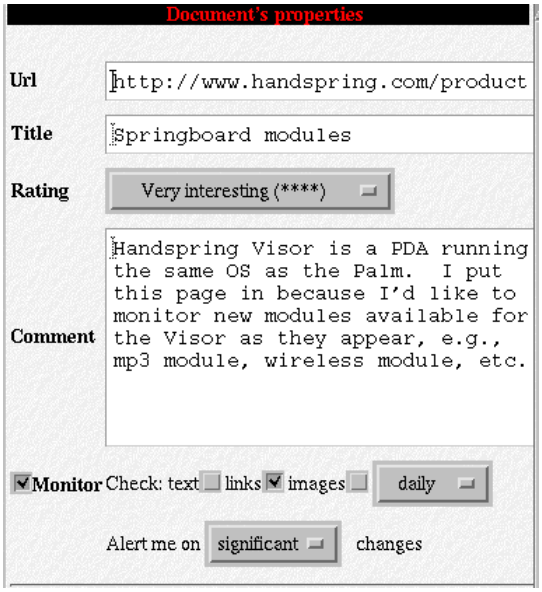


Figure 1. Request to monitor an item for changes

On any day for which the monitoring agent detects at least one changed URL among the set of URLs monitored for the KP member, the system sends out an e-mail listing all the changed URLs detected by the monitoring agent that day for that user. Users can modify their preferences within KP to request not to receive this mailing; the default is to receive it.

We modified the KP interface to contain an additional view showing changed URLs, called the *Updates* Folder, exemplified in Fig. 2. The Updates Folder contains two sections, one containing notifications from the agent, the second containing evaluations by the community as well as by the agent. Clicking on the document title shows a diff view of the URL, highlighting the changes. The Unix *diff* tool is used to highlight visually changes on HTML documents. As *diff* compare lines of text, we transform the HTML by separating HTML tags from actual text on separate lines. By analyzing the *diff* output, the agent is able to insert special signs in the page that show what has been added, what has been updated, and what has been deleted. Modifications in the HTML header are detected and a warning is inserted on top of the document "HTML header updated." This is the case, for example, when the document title is modified. Scripts embedded in HTML document are not currently handled.

In parentheses after the link, there is information concerning the number of users monitoring the URL (its subscribers) and the agent's evaluation of the extent of the change using natural language constructions. The monitoring agent indicates the size of the change and provides indications of whether that change is significant: how much the size of the page has changed, how many links/images/keywords have changed. There is also a form allowing users to easily enter their evaluation of the nature of the change.

Figure 2. Updates Folder

In the community section of the Updates folder, the change notifications are filtered by evaluations coming from others monitoring the same URLs. Changes greater than the user's threshold for the item are flagged. User evaluations are highlighted using color to set them apart from agent evaluations.

Community-evaluated changes display users' evaluations in addition to the monitoring agent's description. A user evaluation consists of a categorization of the importance of the change (*negligible*, *significant* or *important*) and (optionally) a comment on the change. When there are multiple differing user evaluations of the change, the system takes a weighted average of the set of evaluations in order to decide who to notify. For the moment, we have decided to bias the final evaluation, so that an evaluation of *important* has greater weight than *significant* and so on. This means that it will take many ratings of *negligible* to outweigh just one rating of *important*. An alternative would be to learn similarities between users' evaluations over time to predict the importance of a change based on others' evaluations and their similarity to the user. In fact, since in this implementation the monitoring agent is integrated with a recommender system, we

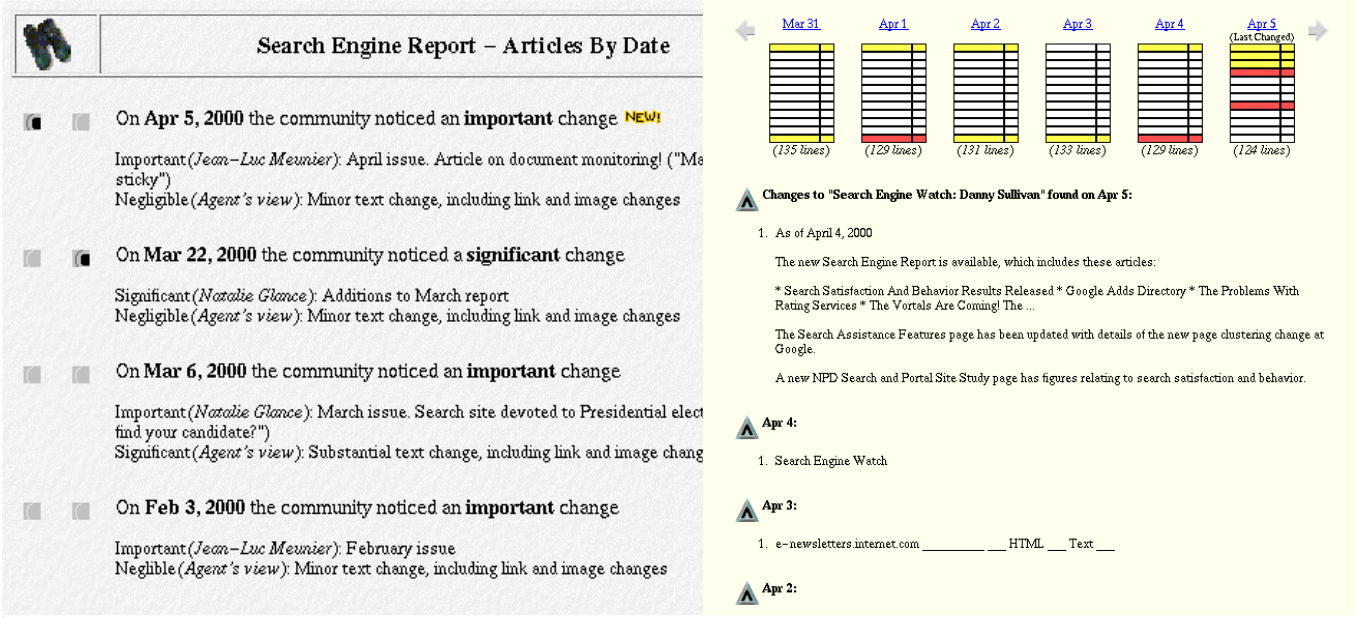


Figure 3. (a) Collaborative version history captures essential changes; (b) Automatic version history is very noisy.

could use the correlation data already calculated by the latter. In addition, machine learning techniques could be applied to improve the performance of the monitoring agent for each user, to learn for example that a negligible change in a URL tends with large enough probability to lead to a judgment of *significant* or greater by the user.

The system puts a little flag next to the *Updates* tag of the folder whenever there is either a newly detected change in a URL the user monitors or a change for which the weighted average of user evaluations is greater than the user's threshold. As a result, users know when there is a reason to check their *Updates* folder.

In addition to the change evaluation by the agent and by the users, there is also a link to the *document history* of the URL. The document history consists of the five most recent changes and of the five most important previous changes, as judged collectively by the users. If less then five such versions exist, then the *document history* is augmented with versions corresponding to changes judged by the agent alone. An example document history is shown in Fig. 3a. The versions are kept in cache and can be retrieved by following the links.

Notice in the *Updates* Folder that following the link to the document history is a natural language comment indicating the number of changes since the user last checked his/her *Updates* Folder (e.g. "There have been several changes, some of which were important since Apr 13, 2000"). Using the document history, the user can then quickly get up-to-date with all changes since his/her last view of the document. Compare this with the summary produced by *DailyDiffs* (Fig. 3b). In this case, only one of the reported changes corresponds to a change likely to be of interest to the user; the others are irrelevant and correspond to ad banner changes.

3.3 System Architecture

In this section we briefly present and explain the architectural and technological choices we made for implementing the integrated system, as well as some of the inner workings of the monitoring agent. From the start, the monitoring agent was conceived as an autonomous entity, able to work independently from other modules. It provides a full-fledged HTML user interface on its own.

When designing the monitoring agent, one of our major concerns was to clearly define the APIs that should be exported to allow the suitable level of interaction with existing modules. Once these were agreed upon and implemented over a network protocol, the monitoring agent could be executed as a separate process (in the UNIX sense), allowing for more robustness (failures do not effect the run-time of other modules) and for straightforward load balancing among hosts. The agent itself relies on a two external services: a Xelda linguistic server and a MySQL database server. Xelda is used to calculate an inverted indices of words and noun phrases from text extracted from the HTML. A fast, scalable and reliable data storage was also needed, as the agent evaluates

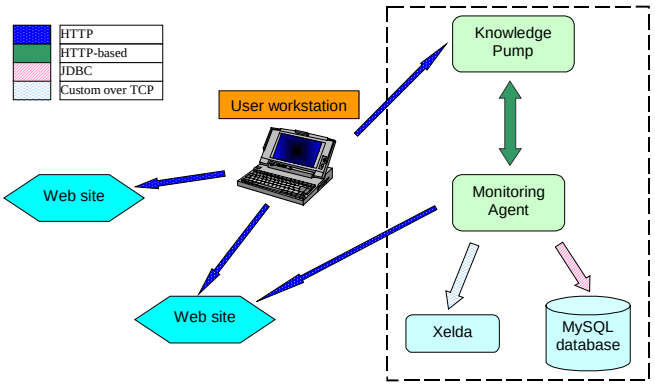


Figure 4. System Architecture

changes by comparing present document content fetched from the Web and a locally stored representation of past document content. The MySQL database engine [13] has already proved to satisfy all those criteria, being in addition to that free for non-commercial use.

We used Java to implement the monitoring agent because: it enforces good object-oriented practices; it provides nice interfacing facilities (particularly in our case, as explained below); and it is fast enough for our purposes.

To get the components to communicate over the network we tried to leverage as much as possible the existing protocols and their implementations, in particular:

- HTTP: the monitoring agent uses HTTP to fetch URL contents from Web servers. Moreover, as Knowledge Pump itself is based on a Servlet-enhanced HTTP server (Apache [3]), this was the technology used to provide the remote API. For communication to the agent, a lightweight HTTP server was embedded on the Monitoring Agent so that it could remotely export its API as well.
- JDBC: the preferred way to talk to SQL databases from Java. We used the MM driver for MySQL [12].
- Custom: the Xelda platform provides a Java implementation of its API using a custom protocol over TCP.

4. USAGE ANALYSIS

We released a new version of Knowledge Pump with the collaborative monitoring functionality in mid-April 2000 to our research center. Knowledge Pump itself was first released in January 1998 and has a solid base of about 20 active contributors with about another 15 active "lurkers" (members who visit items recommended to them, but who rarely, if ever, review or contribute new items). Thirteen members have since chosen to monitor pages: the distribution of pages monitored is shown in Fig. 5 (plus one member monitoring 51 URLs, which falls beyond the scale shown). The distribution has a long tail, as has been observed for the distribution of the number of items reviewed/recommended per member [9]. This kind of distribution is typical of systems where a group of users share information: a minority of the users do a majority of the work [2][10]. In the context of collaborative document monitoring, this means that the majority can keep up-to-date on many changing web sites by free-riding on the good will of a few people who are willing to do the work regardless of the contribution of others.

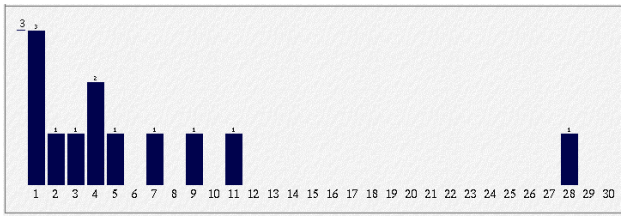


Figure 5. Distribution of #subscriptions per user

In order to truly test the utility of collaborative document monitoring, it would be necessary to work with a larger group of users. However, even with 13 users, we do see some overlap in monitored URLs. See Fig. 6 which shows the distribution of the number of documents monitored by different numbers of people.

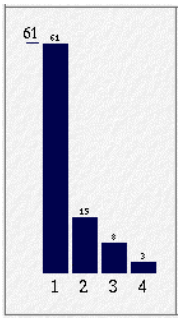
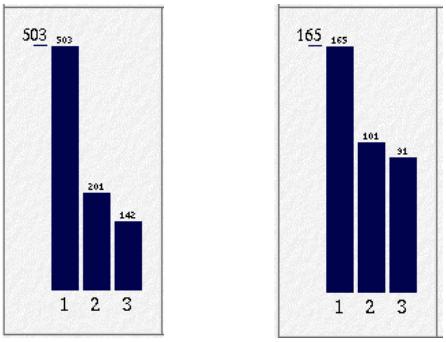


Figure 6. Distribution of #subscriptions per document

About one third are monitored by two or more people. We expect this overlap to grow in a fashion greater than linear as the number of users of the systems increases simply because the overlap in interests in a group of people should depend superlinearly with the size of the group. With only 13 users it is not surprising that almost half of the URLs monitored by at least 2 people have to do with Xerox. (And very few of the URLs monitored by only one person are related to Xerox.)

It is also interesting to compare user evaluations of changes vs. agent evaluations of changes. Fig. 7(a) shows the distribution of negligible (1) vs. significant (2) vs. important (3) changes predicted by the agent. Fig. 7(b) shows the distribution of negligible (1) vs. significant (2) vs. important (3) evaluations by members.



(a) Agent **(b) Human**

Figure 7. Distribution of change evaluations

(1=negligible, 2=significant, 3=important)

Comparing agent predictions and human evaluations we find that the agent agrees with the human 181/357 times (51%). The agent evaluates the change as significant or greater while the human evaluates the change as negligible 39/357 times. The agent evaluates the change as negligible when the human chooses significant or important 86/357 times. Thus, the agent makes a substantive error in judging the nature of the change 125/357 times (35%). This demonstrates that the human input is very valuable at distinguishing significant or important changes vs. negligible changes. Even if the agreement can be improved by improving the monitoring agent (and it certainly can), the value of users' comments on the changes remains.

5. DISCUSSION

In this paper, we have described how a URL monitoring agent can work together with a collaborative system to permit collaborative monitoring of URLs by users. We believe that such an integration will make it much more feasible for users to track Web documents, since the effort will be divided among many. By working with a monitoring agent, a recommender system like Knowledge Pump can effectively handle documents as the dynamic objects they are, instead of as static objects. To our knowledge, Knowledge Pump with a monitoring agent is the first URL recommender system to effectively deal in recommendations of items which change over time.

In addition, we hope that our presentation of the design issues we faced in implementing a collaborative monitoring tool will help others undertaking similar efforts and will encourage discussion that will help us to improve the process for collaborative monitoring that we have put in place.

Future work includes adding more powerful document monitoring methods: monitoring a site or page to a specified depth; capability to monitor pages in a variety of formats, such as PDF, Word, etc., capability to perform monitoring of structured and semi-structured pages. Some of these capabilities are already available in commercial monitoring tools.

Future research issues of interest also include more intelligent notification and filtering of document changes. On the one hand, we hope to improve the ability of the monitoring agent to evaluate changes. On the other hand, we hope to develop more sophisticated algorithms for deciding when to notify users of changes in documents. For example, we could use relevance feedback to train the recommender agent, and correlation among users for deciding when to notify them.

6. ACKNOWLEDGMENTS

The authors thank Cédric Vieau and Nadège Vignol for implementing the first version of the monitoring agent and for debugging and elaborating its specification in collaboration with us.

7. REFERENCES

[1] Ackerman, M., Starr, B., Pazzani, M., “The Do-I-Care Agent: Effective Social Discovery and Filtering on the Web." Proceedings of RIAO'97, 17-31.

[2] Adar, E. and Huberman, B. "Free Riding on Gnutella." First Monday, Oct. 2000.

[3] Apache Web server, The Apache Software Foundation, http://www.apache/org/.

[4] DailyDiffs, http://www.dailydiffs.com/.

[5] Dedieu, Olivier. "Pluxy: un proxy Web dynamiquement extensible." Proceedings of the 1998 NoTeRe colloquium, Oct. 1998, http://www-sor.inria.fr/publi/PPWDE_notere98.html.

[6] Douglis, F., Ball, T., Chen, Y.F., Koutsofios, E., "The AT&T Internet Difference Engine: Tracking and Viewing Changes on the Web," in World Wide Web, 1(1), 27-44, January 1998.

[7] Fishkin, K., Bier, E., "WebTracker - a Web Service for tracking documents," in Proceedings of World Wide Web 6, Santa Clara, California, 1997, http://www.parc.xerox.com/istl/members/fishkin/doc/webtracker.html.

[8] Glance, N., Arregui, D. and Dardenne M., "Knowledge Pump: Supporting the Flow and Use of Knowledge," in Information Technology for Knowledge Management. Eds. U. Borghoff and R. Pareschi, New York: Springer-Verlag, 1998.

[9] Glance, N., Arregui, D. and Dardenne M., "Making recommender systems work for organizations," in Proceedings of PAAM99 (London, UK, April 1999), 1999.

[10] Hardin, G. "The tragedy of the commons." Science 162 (1968), 1243-1248.

[11] Hirschberg, D. S., "Algorithms for the longest subsequence problem," Journal of the ACM, 24(4), 664-675, Oct. 1977.

[12] MM MySQL JDBC driver, http://www.worldserver.com/mm.mysql.

[13] Muslea, I. "Extraction Patterns for Information Extraction Tasks: A Survey," in Proceedings of AAAI'99 Workshop on Machine Learning for Information Extraction (Orlando, Florida, July 1999).

[14] MySQL, T.c.X DataKonsultAB, http://www.tcx.se/.

[15] NetMind, http://www.netmind.com/.

[16] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. "GroupLens: An open architecture for collaborative filtering of netnews," in Proceedings of CSCW'94 (Chapel Hill, NC, October 1994), ACM Press, 175-186.

[17] RMI, Sun, http://java.sun.com/products/jdk/rmi/.

[18] SurfBot, Surflogic LLC., http://www.surflogic.com/.

[19] WebSpector, http://www.illumix.com/.

[20] Xelda, http://www.xrce.xerox.com/ats/.