The Krakatoa Chronicle − An Interactive, Personalized, Newspaper on the Web

Tomonari Kamba
Graphics, Visualization, & Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332−0280
E−mail tomo@cc.gatech.edu

Krishna Bharat
Graphics, Visualization, & Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332−0280
E−mail kb@cc.gatech.edu

Michael C. Albers
School of Industrial & Systems Engineering
Center for Human−Machine Systems Research
Georgia Institute of Technology
Atlanta, GA 30332−0205
E−mail malber@isye.gatech.edu

Abstract:

This paper describes *The Krakatoa Chronicle*, a highly interactive, personalized newspaper on the World Wide Web (WWW). It is intended for Java−aware WWW browsers such as HotJava, and is architecturally quite different from conventional web−based newspapers. Its high interactivity and powerful personalization are the result of sending an interactive agent along with the text of the newspaper to function within the user's web−browser. The agent uses the network connection to the WWW server site to fetch resources dynamically and for updating the user's personal profile through relevance feedback. Users are provided browsing options such as scrolling, maximizing, resizing and peeking at articles. These operations not only enhance the reading experience but also transparently provide information to our agent about the user's locus of interest. Ours is the first web newspaper to attempt a realistic rendering of a newspaper, with a multi−column format and embedded, custom, widgets for easy browsing. We provide dynamic layout control mechanisms that let the user specify how personal and community interests should be interpreted by the agent in designing the layout.

Keywords:

online newspaper, personalization, information retrieval, interactive agent,

active documents, automatic layout

1. Introduction

Although non−commercial usage has dominated the Internet for a long time, some commercial applications have been emerging recently, at a rate which is increasing. According to Outing [Outing 95], "*more than 230 supplemental online services are operated or under development by newspapers worldwide, an increase of about 130% since the end of 1994*". Of these, online newspapers are particularly well suited for the WWW, since the web readily facilitates information retrieval, presentation, and to some extent, layout. However, there are a lot of challenges to be met before they become as pervasive as their printed counterparts. Some of the issues are social and some are technical. While printed newspapers tend to be more portable and easier to manipulate, online newspapers have a powerful argument in their favor − personalization.

Bogart notes [Bogart 89] that personalization has a strong appeal to newspaper readers. Practical considerations have prevented them from being realized under the conventional hard−copy publishing setup. Web newspapers are not subject to the constraints of printed matter. Their reach is equally large and electronic dissemination allows newsfeed to be custom−tailored for individual users. The presentation can be personalized in terms of contents, layout, media (text, graphics, video etc.), advertisement and so on. Newspapers have two important social functions to perform: education and entertainment. Personalization may seem to enhance the latter at the expense of the former. Hence we would need a mechanism to mix in news items with either great popular appeal or high intrinsic value (in the editor’s opinion), into the set of articles that match the user’s interests. To allow multiple perspectives into the same news−feed, the user should have the ability to dynamically affect the way in which the two kinds of articles are combined. This functionality would allow the newspaper to keep pace with changes in reading style as time passes. For instance, a user may want headlines and matters of public interest in the morning, and gradually move to a newspaper composed of articles with personal appeal by evening.

In this paper, we describe an experimental system which implements an interactive, personalized newspaper on the WWW. Basic variables for personalization are computed at the server end, and personal layout happens at the client end, under user control. There are already many online newspapers on the web, such as those accessible from the "Dailies" list [DAILY], and some of them such as CRAYON and Fishwrap provide personalization [CRAYON][FISHWRAP]. However, these have been subject to the limitations of HTML [HTML]. The precise formatting and multi−column layout one is accustomed to in printed newspapers is hard to support. Interactivity is restricted to point and click style interaction and changes take a long time to occur, due to the high, client−server, round−trip latency and restrictive update model. Since HTML is not dynamically extensible and tends to evolve slowly, it is unlikely that the custom widgets needed to program a newspaper will ever be

supported. Hence we needed to turn to a new paradigm, the programmable Java "applet" feature available in Java−aware browsers such as HotJava [JAVA] and (future versions of) NetScape [Netscape].

Java [JAVA] is an object−oriented, programming language which can be used to create architecture−neutral, byte−code programs. A Java applet is a java program designed specifically to be embedded in HTML documents. Java applets can be network−aware applications which users can interact with. A Java−aware browser is a WWW browser that embeds a Java byte−code interpreter and can handle applet tags. Since the downloaded code runs on the client locally, fairly involved computation and interactive, custom−designed, user−interfaces can be supported. In addition, Java has a library for handling TCP/IP protocols and can access remote objects via URLs [URL] easily, which allows us to have continuous bi−directional communication between server and client.

The Krakatoa Chronicle provides some interactive features that other online newspapers do not:

- **Flexible Layout Control.** Users can change the syntactic/semantic layout strategy of the newspaper while browsing. For example, they can change the number of articles on the screen with immediate re−layout. They can also change how their personal interests and other people's interests are combined to decide the layout; personal interests are represented as a *user profile*, which is a mapping from a set of keywords to weights.

- **Implicit and Immediate Reflection of User Interests.** A user's explicit feedback and implcit feedback, derived from observations made by the embedded Java agent, modify the user profile. The agent observes the manner in which the user interacts with the articles in the document, and based on the time spent, the interaction techniques used (e.g. scrolling, peeking at, maximizing, resizing), it tries to estimate the user's interest and conveys it to the user's profile. Users can provide explicit feedback about an article employing that article's attached score bar. They can implement part of their profile explicitly by managing a set of keywords.

In the following sections, we will describe our system architecture and implementation, and also how the personalization strategy adopted in *The Krakatoa Chronicle* can be applied to other interactive applications on the WWW.

2. System Architecture

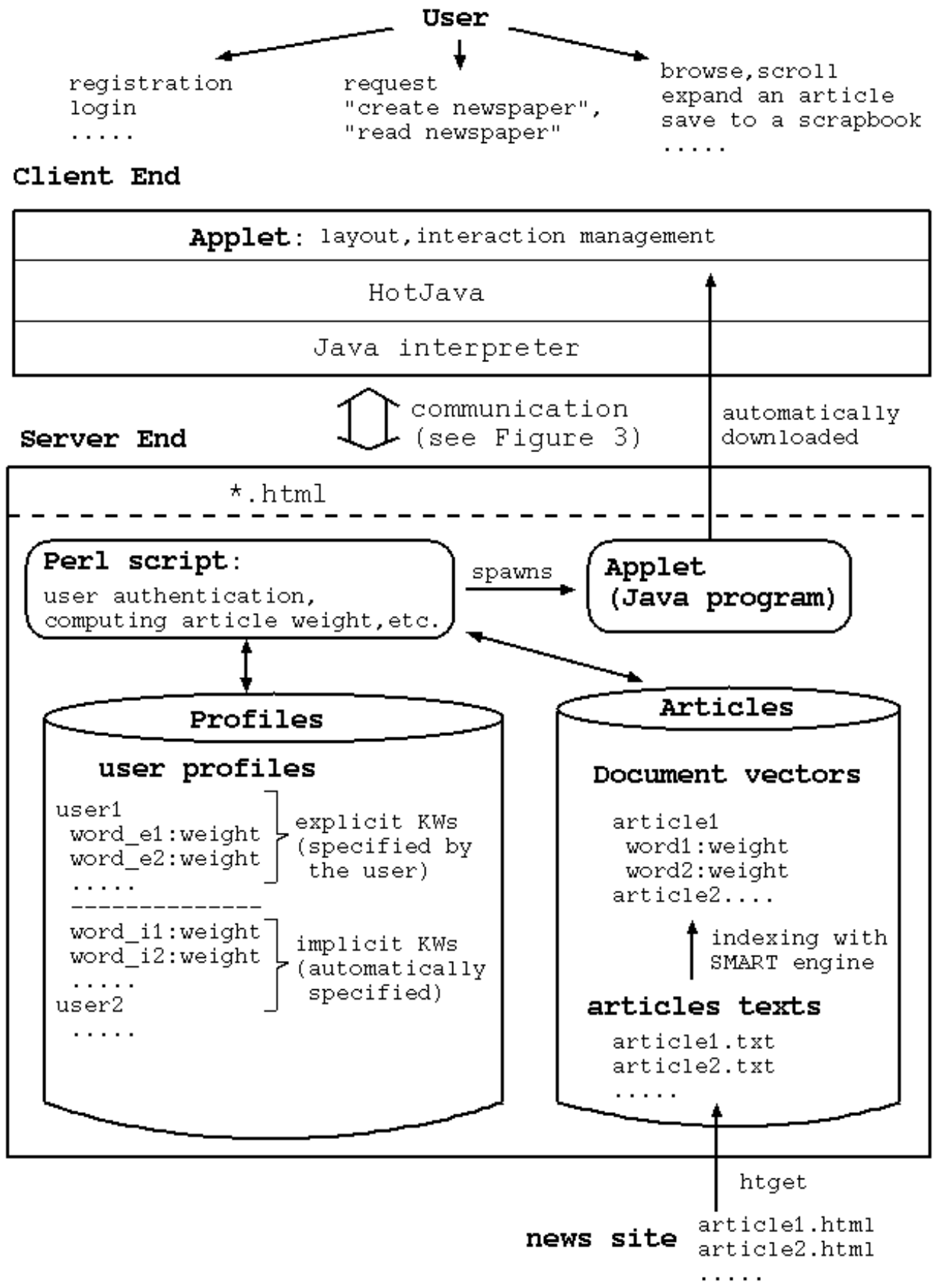

Figure 1. System architecture of *The Krakatoa Chronicle*

Figure 1 shows the system architecture of *The Krakatoa Chronicle*. The system consists of two parts: the server end and the client end.

2.1. The Server End

In addition to user authentication, the server end manages user−profiles, the articles, and their index files, and computes the basic factors needed to decide the layout on the client end.

- **Management of Everyday Articles.** First, articles are collected from several news sites daily (currently we get articles from a single site, the NandO Times [NANDO], with their permission), and changed into plain text for content−analysis and re−formatting. Then, a document vector is computed for each article. Since our research focus is not on indexing, we used a well−known indexing engine, SMART[SMART] to convert articles into document vectors. Articles are converted into representative term−vectors, via the TFIDF (term frequency terms inverse document frequency) metric. A term−vector is a set of keywords and associated weights. The weight of a term shows how good a representative it is of the article within the current set.

- **Management of Profiles.** The server end maintains user profiles. Each user profile has almost the same format as a document vector. The weight of each keyword represents the system's reckoning of the user's interest in the keyword. It is computed when feedback is given. Feedback provides a score for the whole article which is then used to compute scores for individual keywords in its document−vector. Then it is integrated into the user's profile. If a keyword receives both positive and negative feedback it will cease to be a good indicator of the user's interest after a period of time as its cumulative score will go to zero by a process of averaging. Instead if it continuously receives either positive or negative score, it will become a good indicator by virtue of the significant cumulative score it receives. For computing a community score for each article the scores of individual users are averaged over the community. A community is a group of users with similar interests who would like to benefit from each other's preferences [CBN][WEBHOUND][Shardanand 95], but in the simplest case it could include all the users of the system.
  As interaction proceeds, the user's browser provides relevance feedback to the server end, and the user profile is modified.

- **Computing Each Article's Weights.** The server computes each article's weights for a specific user based on how well the article's document vector and the user's profile match. With the SMART engine, we use the personal profile as a query to the document set to compute the weight.

It is worth mentioning that there is no live process at the server end save the HTTP

daemon [HTTP], which spawns cgi−scripts [CGI] written in Perl programming language when necessary. Indexing is done off−line and in batch mode whenever articles are added.

2.2. The Client End

The client end manages user interaction and newspaper layout within the browser. The code needed to drive the presentation and interaction is downloaded when the user accesses the newspaper's web page. Subsequently, the code runs locally and may periodically contact the server site to fetch documents or provide feedback. Since the layout is computed on the client end on the fly, the user can change its strategy flexibly. The user can scroll, peek, maximize, resize or save an article to a scrapbook. When the user does these operations, it is taken to be an indication of the user's interest in the article and gets reflected in the user profile. The user can also explicitly specify his/her personal interest in each article selecting the score on a feedback bar. Details of layout control and interaction will be described later.

3. Creating/Reading a newspaper

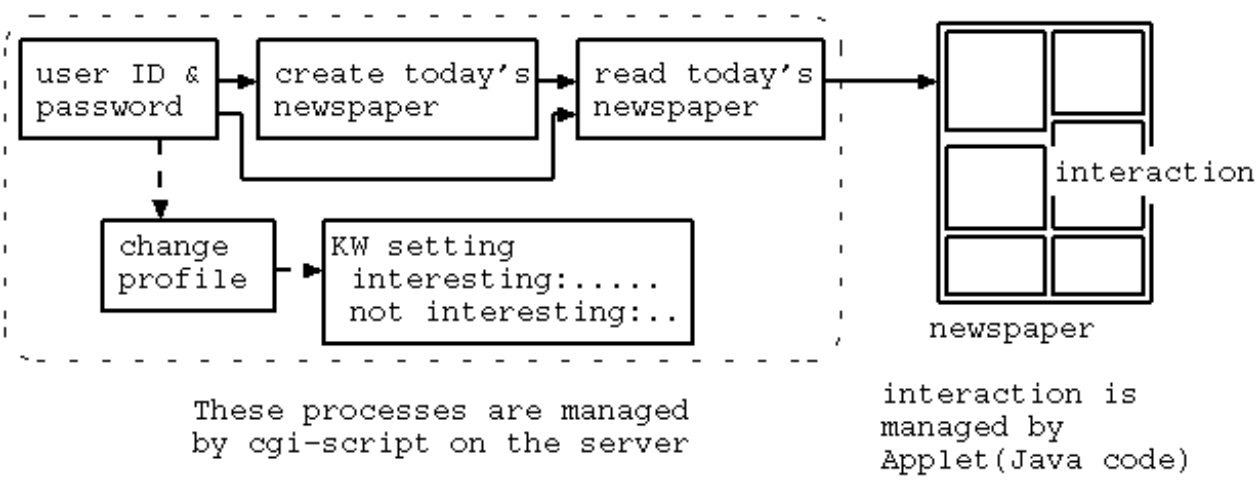

Figure 2. Reading a Newspaper (User's View)

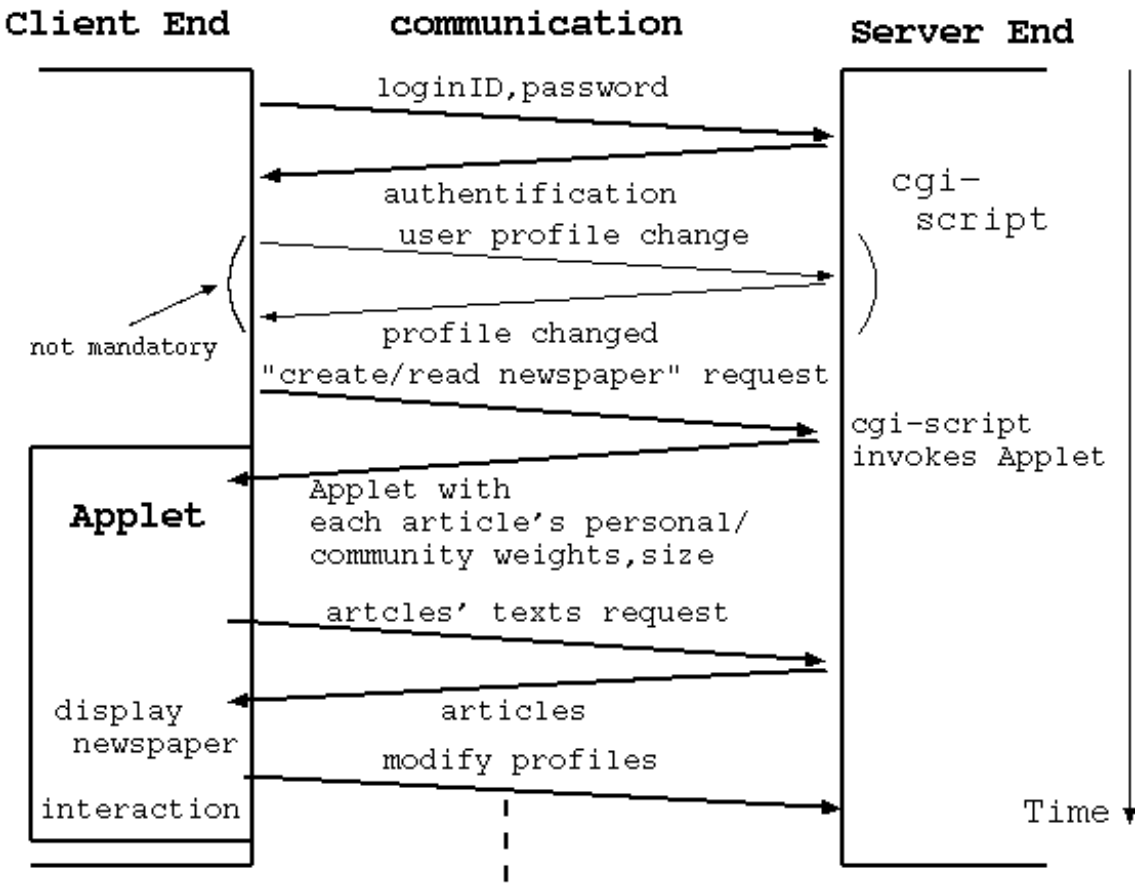

Figure 3. Communication between the Server and the Client

Figure 2 shows the user's view of the process to read his/her personal newspaper, and Figure 3 shows the communication that occurs between the server and the client during the process.

First, the user types in a user ID and a password within a login form. *The Krakatoa Chronicle* is currently available only to a limited number of registered users (most of them are on−campus) because of copyright reasons. After authentication by the cgi−script on the server, the user can select either "Change Profiles", "Create Today's Newspaper", or "Read Today's Newspaper". The "Read Today's Newspaper" option is available only if at least one version of the day's newspaper is present (having been created earlier).

Selecting the "Change Profiles" button is not mandatory, but with this, the user can bootstrap his/her profile before the first newspaper is created. The user can specify topics of interest/disinterest by keywords. These keywords are called "explicitly specified keywords" and are distinguished from "implicitly extracted keywords", which are created and modified while the user is reading a newspaper. The weights for explicitly specified keywords are not modified by implicit feedback. Even if the user doesn't provide feedback explicitly, the system can automatically create and modify personal profiles during the user's reading process by observing the user's actions.

When the user chooses the "Create Today's Newspaper" button, each article's weight is computed and an applet is composed by the cgi−script and sent to the client. The applet computes the layout using each article's weight and other factors, and displays the laid−out newspaper after getting formatted pieces of text from the server. When the user chooses the "Read Today's Newspaper" button, the last computed newspaper is sent over. This allows users to return to the same newspaper even though their profiles have changed.

4. Layout Control and interaction

4.1. Layout control

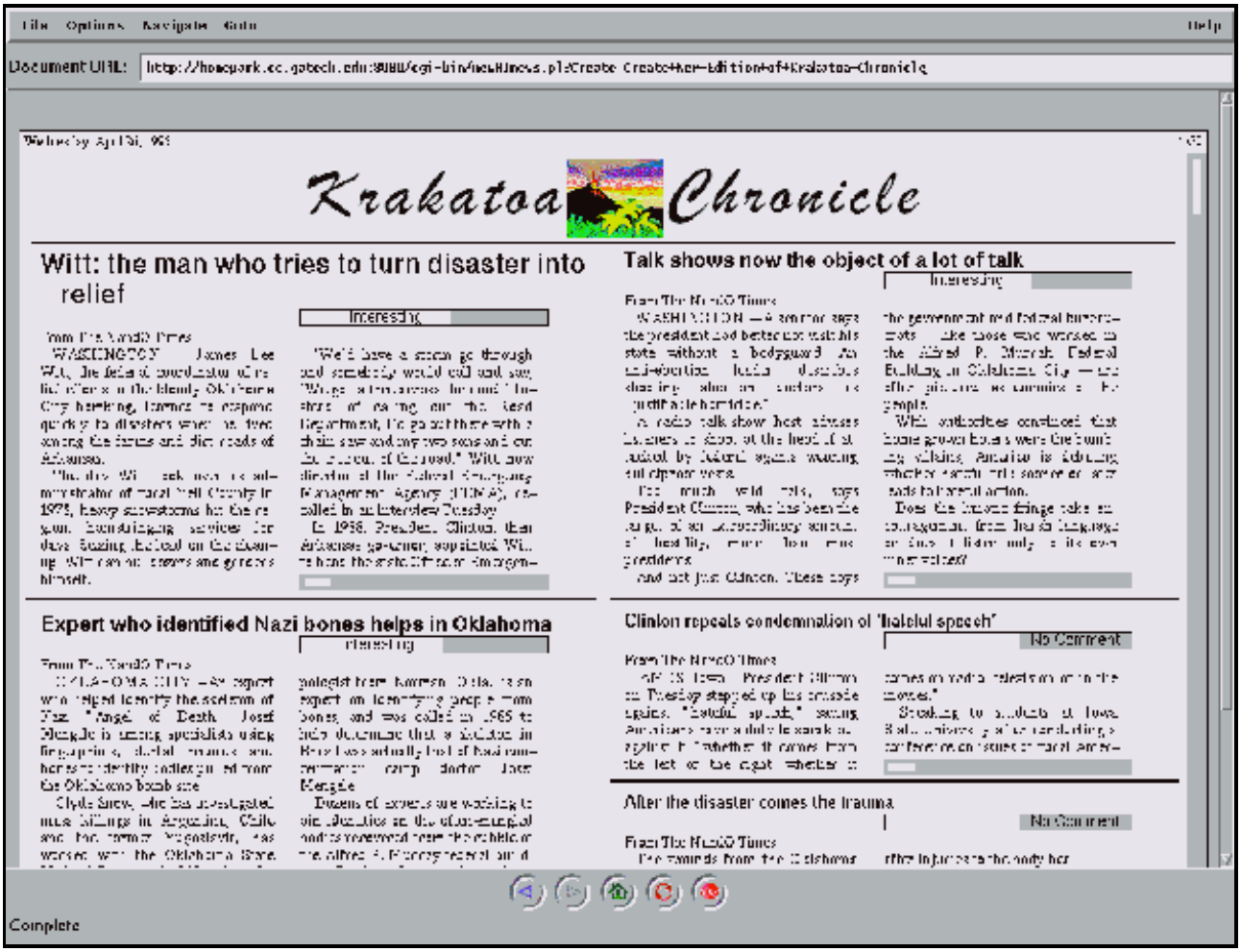

Figure 4. Example of *The Krakatoa Chronicle* Screen

Figure 4 shows a typical screen of *The Krakatoa Chronicle*. It is the first web newspaper to attempt a realistic rendering of a newspaper, with a multi−column format to resemble actual newspapers and embedded widgets for convenient browsing.

We will not discuss the layout algorithm in detail, and only some of the layout

strategies are shown below.

- Articles should be laid out in their entirety whenever possible.
- The title should be placed in a single line preferably.
- Rectangular tesselations of the page are sufficient.
- Pictures may be scaled down to fit in the space available, but will be linked to the original one.

One of the important features that differentiate *The Krakatoa Chronicle* from other WWW−based newspapers is its flexible layout control. Layout is a function of several parameters: score that each article receives based on the user's profile (the user score), the average score received by each article over the community of users (the community score), and also the size and composition (e.g. the number of pictures) of each article. Since there are many ways in which these parameters may be combined to decide the final layout, we give the users a set of controls which they can manipulate to dynamically change the significance of various layout parameters. Specifically the following parameters are controllable (see Figure 5):

- **User Score vs. Community Score.** This decides the ratio in which the scores are combined in deciding the layout. The order of articles (left−top is the most important and right−bottom is the least) for a user is decided by each article's score, and the score is a function of the user's score and the community score:
  final_score = personal_score × r + community_score × (1−r)
  wherein 0 <= r <= 1
  This factor is very important from social point of view. No two people will want the same mix of personal and community content, and a given user may want a different mix at different times.

- **Sensitivity Factor.** This decides how significant the scores are in deciding the space allotted to each article. If this variable is large, important articles will have much more space than not−so−important articles, but if this variable is small, important ones and not−so−important ones will have similar portions of screen real−estate, although an article's importance will still affect its position.

- **Density of Articles per Page.** This decides how many articles will be shown on the page. If this variable is large, all the articles will be shrunk while keeping to the inter−article ratios dictated by the sensitivity factor.

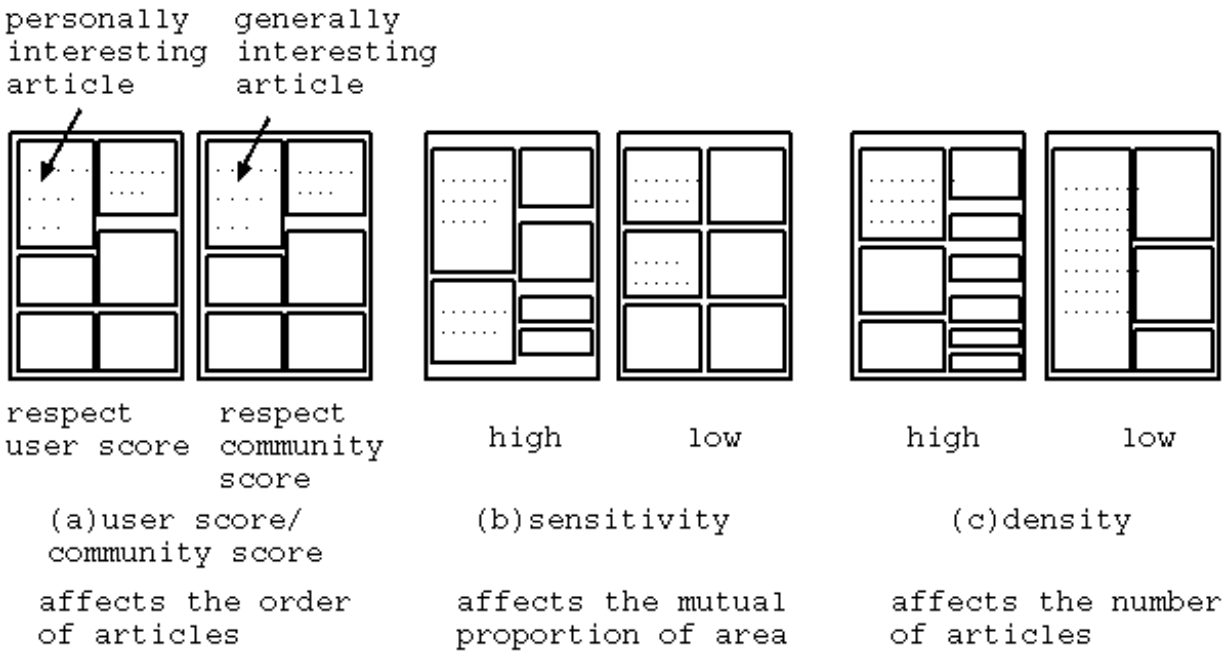

Figure 5. Layout parameters

We hope to let users arrive at a comfortable setting for these controls, over the course of the experiment, and probably learn something in the process. The ability to change layout settings flexibly will allow users to obtain multiple views of the newspaper. We believe these multiple views show users what the community is interested in while allowing for personal customizations via previous user feedback.

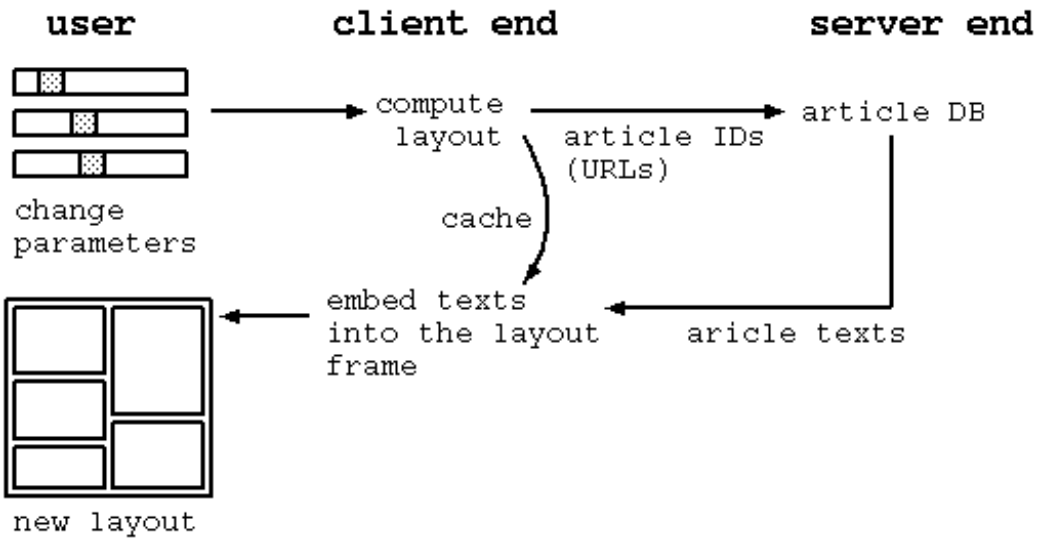

Figure 6. Layout control mechanism

In *The Krakatoa Chronicle*, this flexible layout control was implemented with frequent connections between the applet (Java code) on the client end and the articles on the server end (see Figure 6). Whenever the user changes the setting of layout parameters with sliders, a new layout is computed. Article files are cached, and the agent fetches new articles from the server end only if needed.

4.2. Interaction

The Krakatoa Chronicle provides interaction techniques to support browsing in ways a hard−copy newspaper cannot.

- **Scroll/Automatic scroll.** This slides the article vertically if it is single column, or horizontally if it is multi−column to prevent discontinuities in the flow of text.
- **Peek.** This displays the article over the entire screen as long as the mouse is held down over the peek button in the top left corner.
- **Maximize/Revert.** This is like "Peek" but requires double clicking on the body of the article to switch states.
- **Resize.** The user resizes the article and permanently affects layout.
- **Save to a scrapbook.** This saves the article to a scrapbook, which is a page with links to all the articles that the user saved.

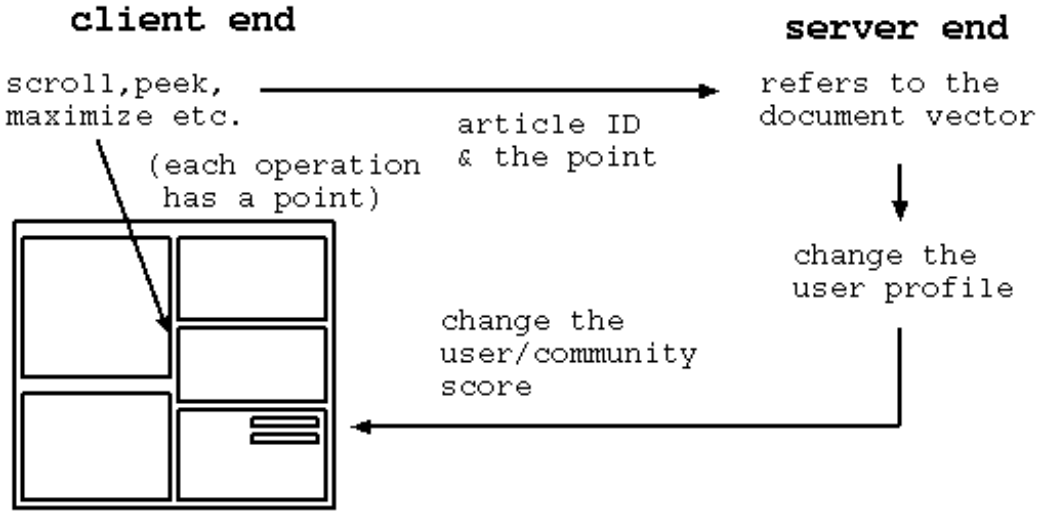

Figure 7. User Operation and Feedback

From the system’s point of view, all these interactions give feedback about the relevance of the article to various degrees (see Figure 7). When the user scrolls, peeks at, maximizes, resizes, or saves an article to a scrapbook, *The Krakatoa Chronicle* increments the user’s interest in the article by a corresponding amount, and subsequently changes the personal profile. There is a score bar beside each article which shows both personal score and community score at the same time, and the user can see changes in the score immediately. The score bar ranges from "Not Relevant" to "Very Relevant" through "No Comment". Initially, this score shows the user’s predicted interest, estimated from the user profile and the article’s document vector.

Once explicit feedback is given by dragging the attached score bar, no further implicit feedback is given for the article. The user’s feedback is taken to be the final word. The community score is the average of all the users of the group that the user

belongs to, and it cannot be controlled directly by the user. This score shows which articles other people are interested in, and will help the user to understand "generally important articles". This score is important in the absence of human editors to assign intrinsic scores to articles.

5. Discussion and Future Work

A set of server end programs fetch each day's articles, translate them into plain text, index them using SMART, and help users register and log on. This is implemented using cgi−scripts written in Perl. Code size of the script to maintain the database and create newspaper is about 500 lines, and the one to handle the login sequence, create new accounts and interact with the user is about 2000 lines. A set of client end programs manage the layout of the newspaper and user interaction. They are written in Java Release 1.0 Alpha3. Code size is about 3000 lines.

We get about 100 articles a day, and the size of each article ranges from 10−200 Kbytes without images. The batch process to index all the articles takes about half an hour and it is done in the wee hours of the morning when the system is unavailable. On a Sun SPARCstation 10 class machine in our Ethernet LAN environment, it takes about 8 seconds to compute each article's score, and approximately 8 seconds to retrieve an article from the server to the client as a single−threaded process. On an initial visit to a newspaper page with six articles, about fifty seconds are required to display the entire page's contents (see Figure 8).

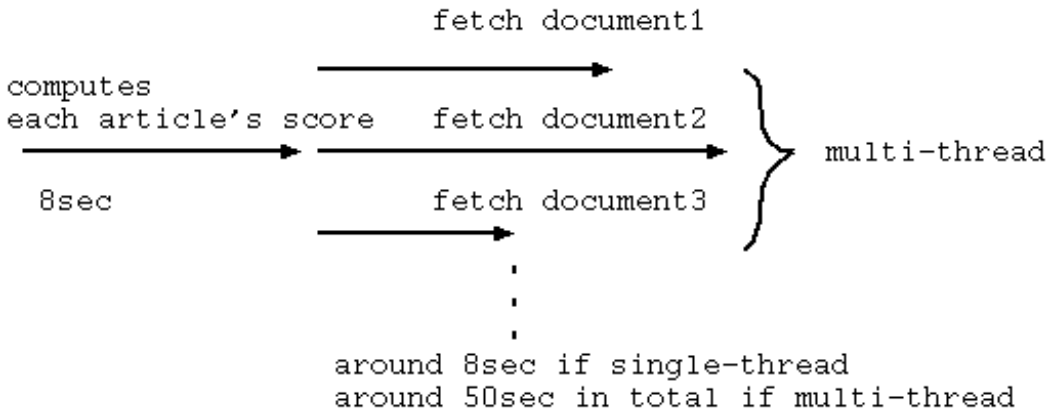

Figure 8. Time to show a newspaper

We have also implemented a version of our personalized online newspaper, which can be shown on conventional (non java−aware) browsers such as Mosaic. Its architecture is similar to other personalizable newspapers on the WWW like Fishwrap [FISHWRAP]. The indexing and user profile/document vector management functions are handled by the same system that implements *The Krakatoa Chronicle*'s backend. However, instead of sending over an applet, it follows the traditional strategy of generating HTML pages for each view into the newspaper, with links to other views. There is no way to dynamically change the layout or get implicit feedback. The newspaper has a listing of articles ordered and font−size coded by

relevance, with links to individual articles. The user can show explicit interest in individual articles, by clicking on a color−coded score bar, attached to the article which modifies the user profile.

Of the many ways to build a user profile, the most common and simplest way is to ask the user to type in keywords that he/she is interested in [YAN 95]. With this method, it is difficult to reflect the user's temporal change in interest or unconscious interest. An indirect way is to ask the user to provide a score for articles he/she reads, and to compute the weights of keywords from the score [Lang 95][Mock]. This method also requires user's conscious involvement which is often annoying. A more subtle way is to consider the time that the user spent on each article, as in prior work based on Internet News [Morita 94], which is said to have worked fairly well. Unfortunately they insisted that subjects devote their entire concentration to the task of reading the article and not take breaks, which is not a very realistic solution. In *The Krakatoa Chronicle*, we included some of these method (typing explicit keywords and/or adding a score to each article). However, the system works even without explicit user involvement. Since anticipated interest values are explicitly represented on the score bars the user will notice that the profile has indicated an erroneous value and will then proceed to give explicit feedback which will correct the profile.

Presently about thirty subjects will start reading *The Krakatoa Chronicle* newspaper daily. After the experiment runs for several weeks, we will reassess the manner in which user actions affect user profiles. Currently, the importance of operations such as "Peek at an Article" is arbitrarily decided. We are also planning to to get statistics on the way people will browse the newspaper, given the choice of interaction techniques they have. In future work, we plan to include dynamic components into a newspaper framework. These would include billboards, live maps, crossword puzzles, shared whiteboards, animated comic strips etc. These are easily implemented and can use a similar scoring/personalization mechanism.

6. Conclusion

We have developed a highly interactive, personalized newspaper on the WWW. It is implemented as an active agent that runs within the web−browser as the newspaper is being displayed. Its main features are realistic rendering, dynamic layout control, interactivity and implicit feedback leading to personalization without conscious user involvement. Information personalization is very important on the WWW, especially for commercial services seeking to provide value−added service for a set of registered users. Our approach to personalization has applicability to other multimedia services on the web as well.

Acknowledgment

We wish to thank the NandO Times who permitted us to use their news articles for our experiment.

Notes

The Krakatoa Chronilce is named after the active volcano, Krakatoa, in the country, Java.

References

[Bogart 89] Leo Bogart, Press and Public: who reads what, when, where, and why in American newspapers, Lawrence Erlbaum Associates, Publishers, 1989.

[CBN] Community−Based Navigation, <URL:http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/HCI/hill/home−page.html>

[CGI] Overview of CGI, <URL:http://www.w3.org/hypertext/WWW/CGI/Overview.html>

[CRAYON] CRAYON,<URL:http://sun.bucknell.edu/~boulter/crayon/>

[DAILY] Dailies:U.S.Newspaper Services on the Internet, <URL:http://marketplace.com/e−papers.list.www/e−papers.us.dailies.html>

[Faloutsos] Christos Faloutsos and Dougtas Oard, A Survey of Information Retrieval and Filtering Method, <URL:http://www.glue.umd.edu/enee/medlab/filter/papers/survey.ps>

[FISHWRAP] Fishwrap, <URL:http://fishwrap.mit.edu/>

[HTML] HyperText Markup Language (HTML): Working and Background Materials, <URL:http//www.w3.org/hypertext/WWW/MarkUp/>

[HTTP] Overview of HTTP, <URL:http://www.w3.org/hypertext/WWW/Protocols/Overview.html>

[JAVA] HotJava Home Page, <URL:http//java.sun.com>(Java and HotJava are trademarks of Sun Microsystems, Inc.)

[Jennings 92] A. Jennings and H. Higuchi, A personal news service based on a user model neural network, IEICE Transactions on Information Systems, 75(2), 198−209, 1992.

[Lang 95] Ken Lang, NewsWeeder: Learning to Filter Netnews, ML95, <URL:http://anther.learning.cs.cmu.edu/ml95.ps>

[Mock] Kenrick J. Mock and V. Rao Vemuri, Adaptive User Models for Intelligent Information Filtering, <URL:http://www.glue.umd.edu/enee/medlab/filter/gwic.ps>

[Morita 94] Masahiro Morita and Yoichi Shinoda, Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval, SIGIR’94, 1994

[Netscape] Netscape to license Sun’s Java programming Language, <URL:http://www.netscape.com/newsref/pr/newsrelease25.html>

[NANDO] The NandO Times, <URL:http://www.nando.net/newsroom/nt/nando.html>

[Outing 95] Online Newspaper Services main menu, MediaInfo Interactive, 1995). <URL:http://marketplace.com/e−papers.list.www/e−papers.home.page.html>

[Shardanand 95] Upendra Shardanand and Patti Maes, Social Information Filtering: Algorithms for Automating "Word of Mouth", Proceedings of CHI’95, 1995. <URL:http://agents.www.media.mit.edu/groups/agents/papers/ringo/chi−95−paper.ps>

[SMART] SMART, <URL:ftp://ftp.cs.cornell.edu/pub/smart>

[URL] A Beginner’s Guide to URLs, <URL:http://www.ncsa.uiuc.edu/demoweb/url−primer.html>

[Yan 95] Tak W. Yan and Hector Garcia−Molina, SIFT − A Tool for Wide−Area Information Dissemination, USENIX Technical Conference, pp. 177−186, 1995. <URL:ftp://db.stanford.edu/pub/sift/sift.ps>

[WEBHOUND] The Webhound WWW Document Filtering System, <URL:http://webhound.www.media.mit.edu/projects/webhound/doc/>