From User Access Patterns to Dynamic Hypertext Linking

Tak Woon Yan
Matthew Jacobsen
Hector Garcia-Molina
Umeshwar Dayal

Abstract:

This paper describes an approach for automatically classifying visitors of a web site according to their access patterns. User access logs are examined to discover clusters of users that exhibit similar information needs; e.g., users that access similar pages. This may result in a better understanding of how users visit the site, and lead to an improved organization of the hypertext documents for navigational convenience. More interestingly, based on what categories an individual user falls into, we can dynamically suggest links for him to navigate. In this paper, we describe the overall design of a system that implements these ideas, and elaborate on the preprocessing, clustering, and dynamic link suggestion tasks. We present some experimental results generated by analyzing the access log of a web site.

Keywords:

Retrieval and Resource Discovery, Dynamic Hypertext Configuration, Data Mining

Introduction

Imagine a shopper browsing through a department store online catalog using his favorite web browser. Suppose he is interested in purchasing clothes for himself, consumer electronics, and sports goods. As these items are sold by different departments, and the pages of these departments may not be linked together directly, he has to navigate through many intervening pages before he can locate the desired ones.

Potentially there may be a whole *category* of users who share similar interests as the "male yuppie" shopper here. There may also be other categories of shoppers, such as expectant mothers, retired adults, or skiers. Suppose the catalog designer has some way of identifying the access pattern for each category of user. To help the users navigate more easily, the designer may link up pages that are identified to be often accessed together. For example, the electronics page may contain a link to the sports goods page.

In this paper, we first show how categories of users can be identified by analyzing user access logs with *clustering* techniques [HA75]. We show that these techniques can discover categories that might not be thought of by the catalog designer beforehand, and might not be inherent in the static hypertext layout (e.g., a user chooses to jump to a page after doing a keyword search). The idea of identifying user patterns applies to accesses to a (single) web site with a multitude of information in general, even though we have described it under the scenario of an online shopping catalog.

Once these common patterns are discovered, they can help in the design of the static hypertext organization (i.e., which pages are linked together), as suggested above. More interestingly, we may customize the organization on-the-fly and dynamically link hypertext pages for individual users. The idea is to try and match an active user’s access pattern with one or more of the categories discovered from the logs. Pages in the matched categories that have not been explored by the user and are not adjacent to the user’s current position may serve as navigational hints for the user to follow. In the example above, after the shopper has accessed the men’s clothings and electronics pages, we may suggest a link to the sports goods page.

Such dynamic linking is desirable for a number of reasons. Firstly, it is customized for each individual user, based on what interests the user has shown so far. A static link (such as a link from the electronics page to the sports goods page) may not be applicable to all users. Secondly, because the content of a web site may keep changing, automatic clustering and dynamic linking provides more up-to-date suggestions than a static design. Finally, as the number of categories may be large, adding suggestion links may become cumbersome for the designer.

Besides dynamic link suggestion, we may put user category information into other uses. For example, it may help in enhancing server performance. The server may prefetch pages that a user is likely to visit soon, based on what he has accessed and what category he falls into.

The contributions of this paper are as follows. We describe the overall design of a system that implements the clustering and dynamic linking ideas. We discuss in detail issues on log-preprocessing, clustering, and dynamic link suggestion and present our solutions. We report some experimental results generated by analyzing the access logs of a web site to support our ideas. Finally, we are distributing the log analyzer as public domain software:

ftp://www-db.stanford.edu/pub/analog/analog.0.1.tar.Z

This tool can be used to help web administrators analyze user access logs generated by a NCSA httpd server [NCSA95].

Related Work

Mining information from large datasets is an area of active research (see, e.g., [FU95]). Clustering algorithms [HA75, BP92] form one class of data mining techniques. In our work, we apply clustering techniques to mine web user access patterns.

Researchers in the hypertext community have studied dynamic hypertext configuration. In one approach [SF91], criteria for reconfiguration are supplied by the hypertext designer. Based on a user’s accesses, these criteria are checked and if satisfied, the linkage among documents adapts in a predefined way. WebWatcher [AFJM95] proposes a learning approach to provide navigation hints. User feedback is used to improve the quality of the hints. Letizia [LI95] records what interests a user has shown, e.g., links followed and keyword searches performed. It then looks ahead in the neighboring pages that might be of interest and suggests them to the user. Our approach of adding "suggestions" to a requested page is borrowed from [AFJM95] and [LI95]. Our proposal of analyzing access logs, finding common patterns, categorizing users, and online matching have not been studied before.

System Design

In this section we give an overview of the system design. Refering to Figure 1, the system consists of three main components: a web server capable of maintaining *user session* information (see below), an offline module responsible for log analysis, and an online module responsible for dynamic link generation.

The web server in our design is just like a typical server that supports HTTP [BFF95], such as NCSA httpd server. The only difference is that it supports in addition the notion of a user session; i.e., an ongoing interaction between the user and the web server. As a user may have different information goals each time he accesses a web site, we believe it is better to model user interests on a per session basis. However, in HTTP, connections between a web client and a server are stateless and there is no notion of session at all. To overcome this difficulty, we may, as others have done (e.g., [TW96]), encode session identifiers in URLs. The first time a user accesses the server, a new session identifier is generated. In the HTML document returned, this identifier is encoded in all URLs refering to objects on the same web site. Thus, the next time the user clicks on these encoded URLs, the session identifier is passed back. This way, a session can be maintained across multiple URL requests. An identifier timeout mechanism can also be used to make sure different sessions from the same client are given different identifiers. We have modified the NCSA httpd server to support all these capabilities.

In the offline module, the preprocessor periodically (e.g., weekly) extracts information from user access logs to generate *records* of users sessions. One record is generated for each session in the logs. The record registers the access patterns exhibited by the user in that session. Records are then clustered into categories, with "similar" sessions put into the same category.

The online module performs dynamic link generation. When a user requests a new page, the module tries to classify his current partial session record against one or more of the categories obtained offline. The top matching categories are identified, and links to unexplored pages contained in these categories are inserted at the top of the page shipped back to the user.

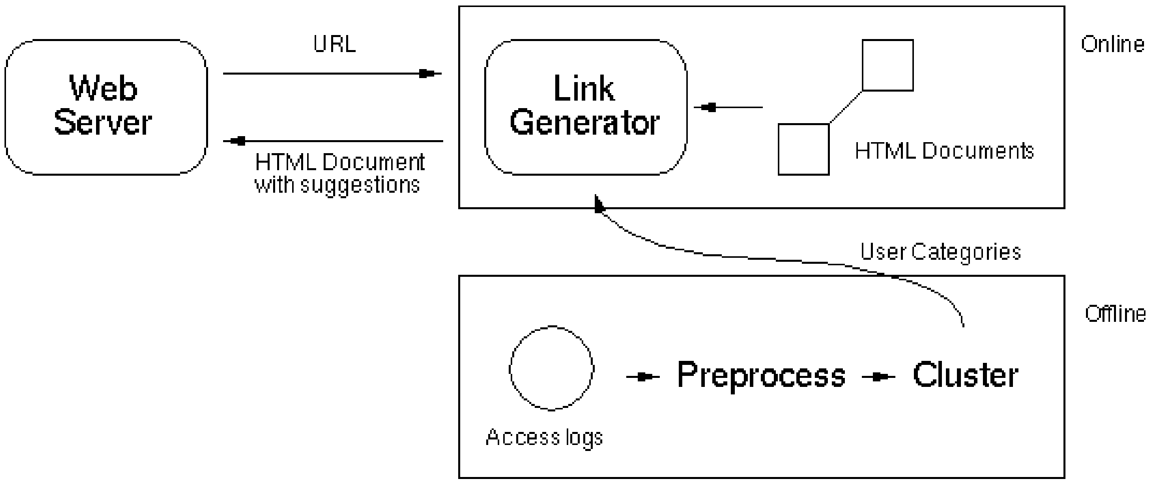

Figure 1. Overview of System Design

In the following we elaborate on three areas in this design: preprocessing, clustering, and dynamic link generation.

Preprocessing

We may view a web site as consisting of a number of *interest items*. For example, we may consider an HTML page as an interest item. An alternative may be to group pages into "semantic" interest items; e.g., all pages on the subject of "professional sports" form one item. Another alternative, applicable in an online catalog scenario, is to consider a purchase item as an interest item. Below we just assume each HTML page is an interest item.

During a session, a user may show varying degrees of interests in these items. If there are *n* interest items in the web site, we may represent a user session as an *n*-dimensional vector, the *i*-th element being the *weight*, or degree of interest, assigned to the *i*-th interest item. If we view an HTML page as an interest item, then we can give it a weight equal to the number of times the page is accessed, or the amount of time the user spends on the page (perhaps normalized by the length of the page), or the number of links the user clicks on that page. We experimented with a number of options and the results are reported below.

Such an *n*-dimensional vector forms a user session record mentioned above. Session vectors that are "close" together in the *n*-dimensional space form a cluster. The task of the preprocessing step is to convert the information in user access logs into the vector representation.

Below we show how a user access log from a web server supporting sessions may look like. It shows four requests from one session. (Requests from other sessions are not shown.)

```
foo.bar.edu - - [16/Nov/1995:18:50:04 -0800] \
        "GET /$$87612/sigmod_record/ HTTP/1.0" 200 1252
foo.bar.edu - - [16/Nov/1995:18:50:14 -0800] \
        "GET /$$87612/sigmod_record/issues.html HTTP/1.0" 200 653
foo.bar.edu - - [16/Nov/1995:18:50:23 -0800] \
        "GET /$$87612/sigmod_record/9-95/ HTTP/1.0" 200 3565
foo.bar.edu - - [16/Nov/1995:18:50:29 -0800] \
        "GET /$$87612/sigmod_record/issues.html HTTP/1.0" 200 653
```

The first line shows the start of the session, originated from a user at foo.bar.edu. The session was assigned identifer $$87612. The page "sigmod_record" was accessed at 18:50:04 on November 16, 1995. The request was successful (return code 200) and the size of the page returned was 1,252 bytes.

Suppose the pages "sigmod_record," "sigmod_record/issues.html," and "sigmod_record/9-95" have been assigned page numbers 200, 135, and 313 respectively. Also assume that we assign page weights by counting how many times a page is accessed. In this case, the above session can be represented by a vector where position 135 has a value of 2, position 200 has a value of 1, position 313 has a 1 value, and all other positions have a zero value. Of course, this vector can be represented more compactly as <(135, 2), (200, 1), (313, 1)>. Note that the page numbers are arbitrarily assigned and do not reflect the order in which the pages were accessed. The order of accesses is an important piece of information, but is not captured by the vector representation presented. We do not address this in this paper.

Clustering

Once the sessions are represented in a vector format, we are ready to run a clustering algorithm against them. The goal of this process is to discover session clusters that exhibit similar interests. When translated to the vector representation, we are interested in finding clusters of session vectors that are "similar." Similarity can be defined in a number of ways. For example, two vectors are similar if the euclidean distance between them is short enough, or the angle between them is small enough.

Clustering (also known as unsupervised learning) is a well-studied area [HA75, BP92] and there are a number of well-known clustering algorithms; e.g., leader, k-means, hierarchical, and fuzzy set approaches. In some algorithms, a vector may belong to more than one cluster, and in that case, cluster membership can be crisp or fuzzy. Interested readers are refered to references such as [HA75] or [BP92]. Our paper presents an approach to apply these techniques to discover useful information in web user access logs.

We may impose a number of constraints desirable for performance (clustering time) reasons or for better clustering outcomes. The first is that we may be interested in only those sessions that access more than a certain number of pages, say *MinNumPages*. For example, it is not very useful to cluster users who just visit the home page and leave. With this constraint we may reduce the number of sessions in our analysis. Secondly, we may be interested only in those clusters that are above a certain size, say *MinClusterSize*. This removes insignificant clusters and may also improve performance.

We illustrate this discussion with a simple algorithm, the leader algorithm (described in [HA75]). The input is a set *V* of vectors. The output is a set *C* of clusters (a cluster is a set of vectors). We start with no clusters and look at the input vectors one by one. For each vector we try to add it to the closest cluster whose median from the vector is shorter than an euclidean distance of *MaxDistance*. If no such cluster exists, the vector forms a new cluster.

```
set C to empty
for each v
if the cardinality of v is greater than MinNumPages
then
    find cluster c in C such that the distance
        between the median of c and v is the minimum
        (set d to this minimum) among all clusters in C
    if the distance d is less than MaxDistance
    then add v to c
    else add {v} to C
for each c in C
    if the size of c is less than MinClusterSize
    then remove c from C
return C
```

The leader algorithm has several drawbacks; most noticeably that it is not invariant under reordering of the vectors. Also, the distance between a vector and the final median of the cluster it belongs to is unbounded. However, one very important strength of the algorithm is that it is fast and memory efficient. It requires only one pass over the data, and the vectors do not need to be stored in memory at all. For these reasons, in the Experiments section we used the leader algorithm. Even with this straightforward clustering algorithm, we were able to discover valuable information from access logs.

After the clusters are found, we may compute the median of each cluster and characterize what the cluster represents. The dominating pages are those with the highest associated weights, and we can thus tell what pages characterize a cluster.

Dynamic Link Generation

As a user navigates through the pages of a web site, we need to keep track of what pages he has accessed and put him into one or more known categories if possible. We may then insert links to interesting pages for him to follow.

To maintain active user session information, user access logs are temporarily buffered in main memory. (We use a high performance memory-resident database management system, Smallbase [HP95], for this purpose.) The active session information is maintained using the same type of vectors as in the preprocessing step.

When the online user requests a new URL, the vector is updated. Note that at this point, the vector only represents a partial record of this ongoing session -- there are more accesses to follow. When classifying the partial session vector, the distance between a cluster median and the partial vector may not be a good matching measure, as it is expected the partial vector has fewer non-zero elements than the median vector. An alternative is to count the number of pages the user has accessed in each category. If the count is above a certain predefined threshold (say 2 pages), then a matching category is found.

After all matching categories are identified, we can look at the pages in those categories. Pages that the user has not accessed so far, and are not accessible from the URL just requested, are included as suggestions at the top of the HTML document shipped back to the user.

To illustrate, suppose we found offline a cluster *c* of users accessing the pages on men’s clothings, consumer electronics, and sports goods. Now suppose a shopper who has accessed men’s clothings page is requesting the URL for the electronics page. At this point, the active session of the user is updated to show he has accessed these two pages. The system subsequently matches this session with cluster *c* and includes a link to the sports goods page at the top of the electronics page for him dynamically.

Experiments

To validate our conjecture that clusters exist in user accesses to web sites, and that there are clusters not directly reflecting the physical hypertext structure, we carried out a number of experiments. The logs were taken from the Stanford Database Group web site, at URL http://www-db.stanford.edu. The site hosts a variety of information, including materials on 12 projects, the home pages of 41 group members, member publications, and database course information. It is also the host for the ACM SIGMOD Record Online issues. The logs covered a period of two months, from November 16, 1995 to January 15, 1996. There were 71,642 logged requests in total.

The logs that we were able to obtain did not contain session identifier information. To approximate a session, we considered that requests coming from the same host formed a session. And if an access originating from the same host came after an idle time of more than 24 hours, we considered that the start of a new session. This way, the preprocessing identified 13,240 user sessions accessing 3,984 distinct URLs corresponding to HTML pages. This definition of session is admittedly rough, since for some hosts there could be more than one user, giving rise to some "false sessions" consisting of accesses from more than one user. Note that these false sessions would work against our clustering attempts, since it was unlikely that the aggregated behavior of users from one host would be similar to that of another host. However, when inspecting the logs, we found that most of the accesses came from "small" hosts, rather than large internet service providers such as America Online. Thus we believe the number of false sessions was not high. The fact that we were able to find good clustering outcomes in the experiments reported below supports this claim.

We first did a preliminary experiment to understand user accesses better. Let us call an HTML page request a hit. We plotted the distribution of the hit duration, i.e., how much time a user spends on a page. Figure 2 shows the results. (The last hit of a session was ignored, since we had no way of knowing how long it lasted. Besides, we were just interested in the *distribution* of the hit durations, so not counting the last hit should not affect the distribution.) Note the logarithmic scale of the x- and y-axes. The apparent horizontal lines are actually discrete data points for different x values. The logarithmic scale crowds them together.) The distribution follows roughly the well-known Zipfian distribution; most of the hits are very short. The wide range of the times on the x-axis indicates that using the time spent on a page as the weight given to the page in the vector representation may not be a good idea; one long access may completely obscure the importance of the other pages accessed. We thus decided to use the number of times a page is accessed per session as the weight assigned to that page.

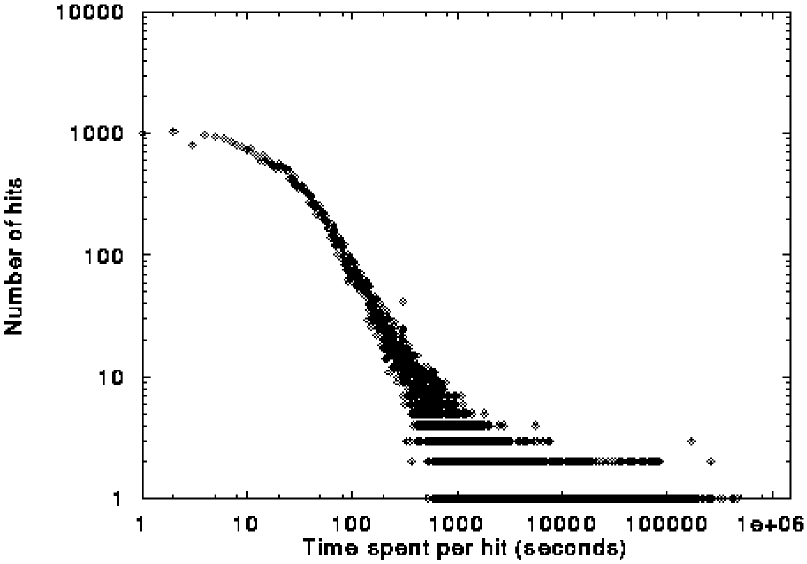

Figure 2. Number of Hits vs. Hit duration

Next, to help us decide on an appropriate value for *MinNumPages*, i.e., the minimum number of pages in a user session for it to be considered in our clustering step, we plotted the distribution of the number of pages accessed in a session. Figure 3 shows the results.

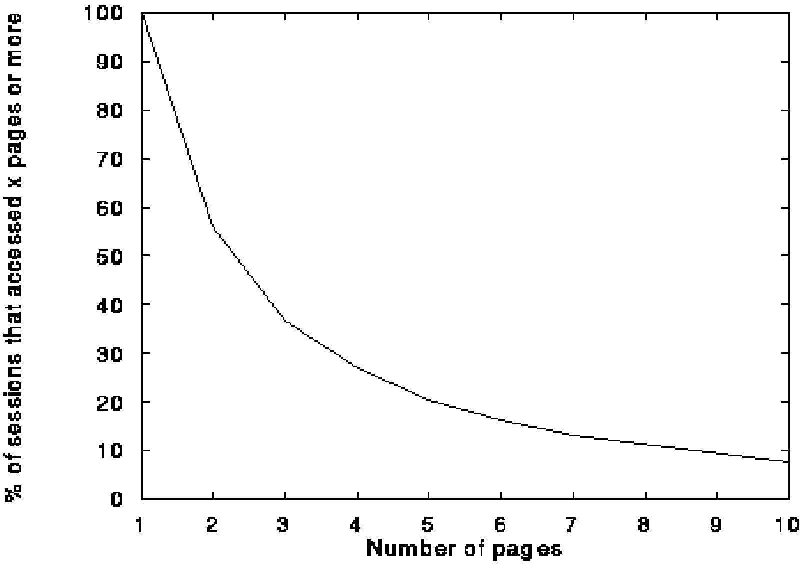

Figure 3. Percentage of Sessions vs. Number of Pages Accessed in a Session

Only about one half of the user sessions accessed 2 pages or more, about 20% accessed 5 pages or more, and less than 10% accessed 10 pages or more. As we were not interested in sessions that were too short, and at the same time we wanted to cover a good portion of users, we decided a value of 5 would be a good choice for *MinNumPages*. This corresponded to 2,709 user sessions.

For the other two parameters *MaxDistance* and *MinClusterSize*, we set them to what we believed reasonable values of 3 and 5 respectively. We ran the leader algorithm against the dataset using this base setting of parameters. We found 41 clusters of size greater than 5. The number of sessions that fell into any one of the clusters was 1,279, which represented *half* of all sessions considered. Thus, our claim that user access patterns can be clustered was validated.

We looked at the pages that characterize each cluster manually. Not surprisingly we found clusters that accessed pages physically linked together. Some of them accessed pages on a project, its members, and the associated publications. Other accessed course information. More interestingly, we also found a number of clusters that were not apparent from looking at the hypertext layout. One large cluster (made up of 66 sessions) was a cluster that accessed pages on object-oriented database systems (the pages are not physically linked together). Two clusters (of sizes 16 and 8) accessed group members of certain nationality. One cluster (sized 10) accessed group members who are alumni of the same university. Another (6 sessions) accessed pages on the topic "information finding."

We also ran a number of experiments that varied the values of the three parameters *MinNumPages*, *MaxDistance*, and *MinClusterSize*. Figure 4 below shows the results of one interesting case in which we varied *MaxDistance*. (Other results are reported in [YJGD96].) In the graph, we plotted the number of clusters with at least five vectors (let us call these *admissible* clusters) against *MaxDistance*. We note that with very short *MaxDistance*, there are many small clusters (each vector would form a cluster by itself if *MaxDistance* were zero). So the number of admissible clusters is small (equal to 2 for a *MaxDistance* of 1). As *MaxDistance* increases, the small clusters merge to become admissible clusters. The number of admissible clusters reaches a maximum when *MaxDistance* is 3. Then, as *MaxDistance* is relaxed further, more clusters merge, and the number of clusters decreases.

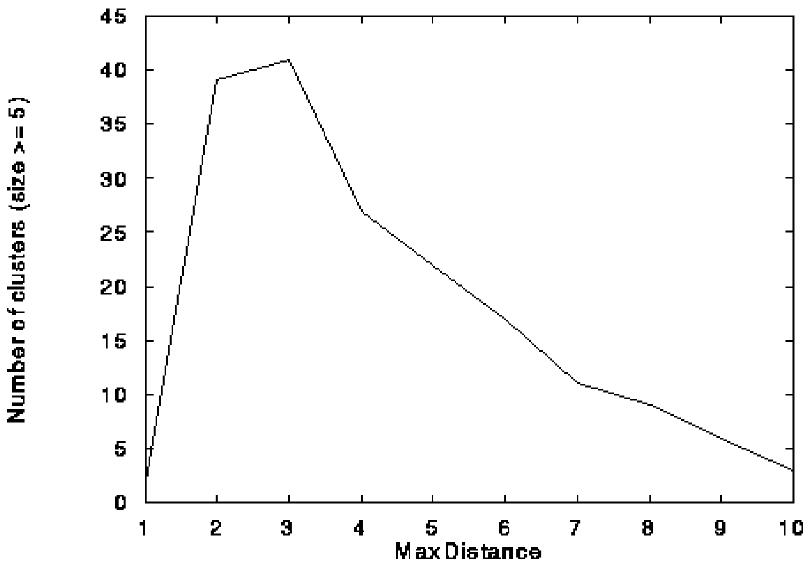

Figure 4. Number of Clusters vs. *MaxDistance*

Finally we measured the times needed to run the clustering algorithm against the dataset. With both *MinNumPages* and *MinClusterSize* set to 5, and *MaxDistance* varying from 1 to 10, the running times were between 33 to 60 seconds on a DEC Alpha workstation. The log-preprocessing step took comparable amounts of time. This was quite efficient for 71,642 user accesses. As both the preprocessing and the clustering (using the leader algorithm) steps require running times approximately linear to the log size, we are confident that the system can cope with logs of larger sizes.

Conclusions and Future Work

We have presented a system design that facilitates the analysis of past user access patterns to discover common user access behavior. This information can then be used to improve the static hypertext structure, or to dynamically insert links to web pages. We have implemented the offline module and the session-logging web server, and started work on the online module. We are distributing the offline module as public domain software:

ftp://www-db.stanford.edu/pub/analog/analog.0.1.tar.Z

Web administrators may find the tool useful for analyzing user access logs generated by a NCSA httpd server.

Our experimental results obtained by analyzing real user access logs show that indeed clusters of user access patterns exist. Further, some of these clusters are not apparent from the physical linkage of the pages, and thus would not be identified without looking at the logs.

For future work, we will look into how to capture the order of accesses to better represent user interests, the use of semantic information to model user interests, the impact of different clustering algorithms on the quality of the cluster information, and the effectiveness of the suggestions given to the users (i.e., we need to evaluate whether the users find the suggestions useful).

References

[AFJM95] R. Armstrong, D. Freitag, T. Joachims, and T. Michell. WebWatcher: a learning appretice for the World-Wide Web. In 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments. March 1995.

[BFF95] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol -- HTTP/1.0. Internet Draft. http://www.w3.org/pub/WWW/Protocols/HTTP1.0/draft-ietf-http-spec.html. October 1995.

[FU95] U. Fayyad and R. Uthurusamy. Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95). AAAI Press. 1995.

[HA75] J. Hartigan. Clustering Algorithms. John Wiley. 1975.

[HP95] Hewlett-Packard. Smallbase Reference Manual. 1995.

[LI95] H. Lieberman. Letizia: an agent that assists web browsing. International Joint Conference on Artificial Intelligence. August 1995.

[NCSA95] National Center for Supercomputing Applications. NCSA httpd. http://hoohoo.ncsa.uiuc.edu/docs/Overview.html. 1995.

[SF91] P. Stotts and R. Furuta. Dynamic adaptation of hypertext structure. In Third ACM Conference on Hypertext Proceedings. Assocation of Computing Machinery. 1991.

[TW96] Time Warner. Pathfinder. http://www.pathfinder.com. 1996.

[YJGD96] T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In preparation. Hewlett-Packard Laboratories Technical Memo. 1996.

About the Authors

Tak W. Yan
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94303
tyan@hpl.hp.com

Matthew Jacobsen
Department of Computer Science
Stanford, CA 94305
jake@db.stanford.edu

Umeshwar Dayal
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94303

dayal@hpl.hp.com

Hector Garcia-Molina
Department of Computer Science
Stanford, CA 94305
hector@db.stanford.edu