# WebFrame: In Pursuit of Computationally and Cognitively Efficient Web Mining

Tong Zheng, Yonghe Niu, and Randy Goebel

Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
Facsimile (780) 492-6393
{tongz, yonghe, goebel}@cs.ualberta.ca

**Abstract.** The goal of web mining is relatively simple: provide both computationally *and* cognitively efficient methods for improving the value of information to users of the WWW. The need for computational efficiency is well-recognized by the data mining community, which sprung from the database community concern for efficient manipulation of large datasets. The motivation for cognitive efficiency is more elusive but at least as important. In as much as cognitive efficiency can be informally construed as ease of understanding, then what is important is any tool or technique that presents cognitively manageable abstractions of large datasets.

We present our initial development of a framework for gathering, analyzing, and redeploying web data. Not dissimilar to conventional data mining, the general idea is that good use of web data first requires the careful selection of data (both usage and content data), the deployment of appropriate learning methods, and the evaluation of the results of applying the results of learning in a web application. Our framework includes tools for building, using, and visualizing web abstractions.

We present an example of the deployment of our framework to navigation improvement. The abstractions we develop are called Navigation Compression Models (NCMs), and we show a method for creating them, using them, and visualizing them to aid in their understanding.

Keywords: data mining, web mining, navigation compression, visualization

## 1 Introduction

After only half a decade, the world wide web (WWW) has become an information playground where every possible learning technique is of potential value for improving web usage — if only one could match application performance goals with appropriate learning technologies. Given enough resources, one can typically find almost anything on the web. In fact, at the estimated growth rate of about 14 million pages per day[1], it is a practical tautology that we can't find value without creating relevant human-oriented abstractions.

[1] Whatever a *page* is?

M.-S. Chen, P.S. Yu, and B. Liu (Eds.): PAKDD 2002, LNAI 2336, pp. 264–275, 2002.

The process of web mining is to create abstractions, with the overall goal of providing both computationally *and* cognitively efficient methods for improving the value of information for WWW users. The need for computational efficiency is well-recognized by the data mining community, which sprung from the database community concern for efficient manipulation of large datasets.

In as much as cognitive efficiency can be informally construed as ease of understanding, then what is important is any tool or technique that presents cognitively manageable abstractions of large datasets. The visualization of web space is based on exactly this idea: that some abstracted form of a large data set can provide insight into some important attributes of that space (e.g., see [6]).

The idea of *web mining* is to apply the tools and techniques of data mining to world wide web (WWW or web) data to induce "interesting" hypotheses that can be used to improve various web usage applications. So the only realistic research direction is to develop web mining software architectures that explicitly address both aspects: computational efficiency in order to provide access to large volumes of data, and cognitive efficiency in enabling users to guide learning processes to information abstractions of appropriate relevance.

The most common instance of this combination is the application of learning to user generated web usage data, sometimes referred to as *web usage mining* (WUM). Here we use WUM as a specific instance of a web mining task, to illustrate the development of a general framework for web mining.

The biggest challenge is to provide a "mining" software architecture that not only provides a harness for efficient learning methods, but also aids in the incremental user formulation of mining goals. This is important because humans are the ultimate determiner of what "relevance" means.

Everyone has their idea of what an abstraction should be. For example, web search engines are a dynamically created operational abstraction that continually update indices, which are then coupled with user query systems to identify relevant web information. Similarly, meta search engines provide another level of abstraction, working at a granularity above search engines by transforming single user queries into several queries to regular search engines. In the other direction, corporate, intranet, and e-business search engines provide local indexing structure, imposing more rigid abstractions that are targeted to circumscribe corporate policies, workflow constraints, and sales strategies.

This is why the notion of *web data* is as broad as the potential applications of its mining. Most current applications of web mining (e.g., [3, 7, 9, 10, 11, 14]) have concentrated on data that is created by the browsing user, beginning with the usage logs produced by web servers (e.g., [12]). Of course there is a broad spectrum of such user "web data," including ordinary web logs, cookies, page exit surveys, search query collections, and even hand collected user surveys. Even though this spectrum of information can itself be incredibly broad, there is another aspect of web data that is even broader and deeper: the web content itself.

The *Web Mining Framework* proposed here is designed to facilitate all aspects of web mining we can currently envisage, including the use of browsing data,

web content, and web meta content. The framework consists of three broad components: 1) data capture tools, 2) learning tools, and 3) evaluation tools.

Our development of this framework is itself an experiment, based on our belief that we need such a framework to assess the various combinations of data, learning, and application evaluation methods. We hope to incrementally improve our framework, and develop answers to questions like "What are the tradeoffs between intrusive data gathering and navigation improvement?" or "Can we measure how the analysis of search queries be focused on specific demographic groups to increase relevance?"

In what follows, we provide a more detailed description of the current status of each of our components, together with examples of our preliminary experiments. In all cases we attempt to be as general as possible in identifying the "inductive opportunities" that arise within web data, and anticipate their ultimate role in improving the value of a range of web activities.

## 2 System Architecture for Web Mining Framework

Like existing data mining architectural proposals (e.g., [3, 7, 11]), the gross level component architecture will require a module to support data capture, a module to support the specification and deployment of a repertoire of learning methods, and, perhaps less common, an explicit module designed to support evaluation of any combination of the first two.

In our particular instance, we have already extended the simple three component architecture into something more elaborate, as depicted by the diagram of Figure 1. One simple way to understand this instance of our architecture is to consider a high-level description of the process control within it. With respect to the diagram of Figure 1, our current web mining procedure can be described as follows:

1. Determine what data can be used, and obtain the data through *Data Acquisition* modules.
2. Convert the original data into specific formats that can be used by various data mining methods. This work is done through *Data Preparation* modules.
3. Determine what data mining algorithms are appropriate for the task. Then, apply the algorithms to the formatted data to obtain corresponding knowledge. This work is done through *Data Mining* modules.
4. The knowledge can be queried or evaluated through various methods. This work is done through *Knowledge Analysis* modules. The evaluation of the knowledge can then be used as a feedback for steps 1-3.
5. We can make use of the knowledge to achieve various tasks in certain applications, and evaluate the performance improvement. This work is done through *Applications & Evaluation* modules. The evaluation of the performance improvement can be used as a feedback for steps 1-3.
6. Original data, formatted data, and even the knowledge can all be visualized through *Visualization* modules, to provide feedback for steps 1-3.

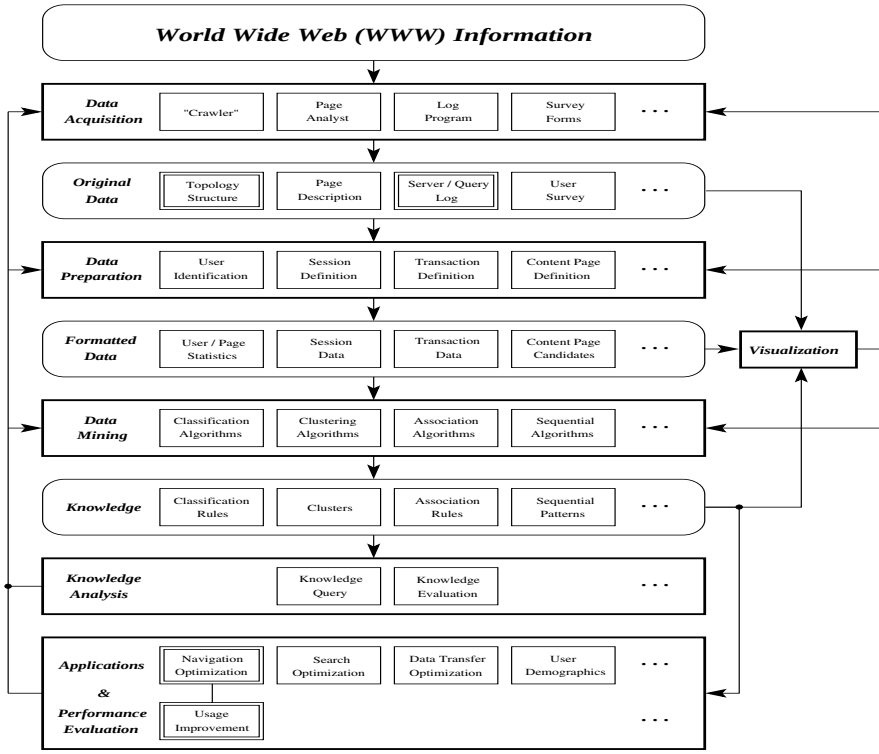



**Fig. 1.** System Architecture for Web Mining Framework

## 3 An Architecture Instance: Creating NCMs

Our first experiments with our web mining framework have the goal of improving user navigation. Our input is web usage data, e.g., web logs, and our evaluation methods (described below) help indicate whether we are actually "improving" a user's navigation, e.g., by helping reduce the number of links traversed to find "interesting" pages.

The missing middle component is that object to be created by various learning algorithms, and then tested to see whether the learning algorithm has found something "interesting" which can provide navigation improvements, as measured by the evaluation methods. We call the objects created by the application of learning methods navigation compression models (NCMs). NCMs are simply some representation of an abstract navigation space, determined as a function of a collection of user navigation paths on actual websites.

## 4 Visualization

There has been a lot of recent work on various forms of data visualization, with a focus on exploiting human cognitive abilities to understand the structure, features, patterns and trends in data [4]. Much of this work derives from the innovative and creative work on graphics design, especially the land mark texts of Tufte [13].

Most existing visualization tools concentrate on either web site visualization (e.g., [8]) or web usage data visualization (e.g., [1]). Because we want our visualization tool to provide insight into both web site structure *and* web usage, we have developed a tool that provides both. Our current version provides the ability to view individual web sites at different levels of granularity, and allows both the static and dynamic display of individual and aggregate user behavior.

Our tool is loosely based on various two dimensional displace techniques that focus visual attention in two space on a single URL, from which links are radially drawn outward. In particular, we use the radial tree algorithm of [2] to provide a two dimensional display of an arbitrary URL. The reason for limiting ourselves to two instead of three dimensions (cf. the three dimensional hypergraphs [8]) is that three dimensional displays require us to solve occlusion problems. Since our goal is a general visualization of both structure and usage, we take this decision to help reduce the complexity of our experimental visualizations.

A screen shot of our WV tool is given in Figure 2. The radial tree is a hierarchical acyclic tree, with each level of the hierarchy represented by a concentric ring of page nodes, distributed around a central focus node. The focus node is designated by the user as a starting URL, which provides the initial focus for any visualization display.

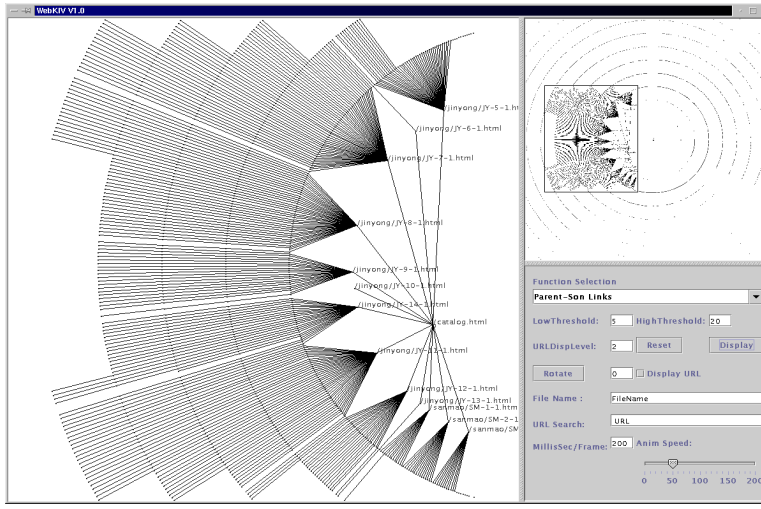



**Fig. 2.** Web Visualization Tool

Similar to heights on a topographic map, nodes linked from the focus URL are displayed on different concentric circles according to their levels in the hierarchical tree. A breadth-first search that avoids cycles is used to traverse the web site, to build the hierarchical structure. The radial pattern of links from a given node is represented by a radial distribution of nodes on a concentric circle, proportional to the number of nodes. Note, for example the concentric rings of linked nodes in the diagram of Figure 2.

This two dimensional display based on concentric link "isobars" doesn't provide any insight to web structure, without an ability to shift the focus and level of detail. With this in mind, our visualization also provides the user with the ability to drag a rectangle over the visualization, then zoom on that rectangle. As shown in Figure 2, a smaller window (context window) always provides a context for the web site structure, while the other larger window (focus window) displays the detail of choice, which is a subset of the larger context. Whenever there is a need to drill down, the selected sub area is enlarged, and both windows are appropriately modified. If the user "zooms in" to sufficient detail then individual URLs are used to label each node.

If we can visualize *individual* user navigation paths superimposed on top of the web site visualization described above, we can begin to recognize well-traversed paths. These might be links that are popular over some particular time period, or trajectories of heavily visited web pages which help us understand how users arrived at web site "hot spots."

In addition to visualizing the navigation paths of individual users, we can also visualize aggregate paths. For example, when an individual traversal of a hyperlink is indicated by drawing a line from one node to another, the aggregate behavior of two users can simply annotate that link for each traversal. There are an arbitrary number of ways to visualize aggregate link traversal (e.g., line width, color, annotation). But when we can visualize *aggregate* user navigation pathers superimposed on top of the web site visualization, we can begin to recognize web navigation clusters in order to understand aggregate user behavior. For example, this is useful to validate web site design (cf. [5]), by statically viewing the distribution of aggregate navigation paths on a web site. And with appropriate navigation path annotation, we can dynamically observe aggregate behavior, e.g., aggregate navigation path changes over different time periods.

## 5 Experiments with Navigation Compression Models

*Navigation optimization* means improving the ease with which users can reach the contents of interest more quickly. The basic idea is that we can discover navigation patterns from previous visitations, and then use these patterns to provide guidance for new users. As explained above, we call our "patterns" navigation compression models or "NCMs."

We can evaluate our NCMs statically by measuring overall navigation improvements. To actually use such information, we can use one of two approaches: one is using synthetic static index pages, each of which contains indices to a set

of pages belonging to similar or related topics; another is using dynamic recommendations, which use NCMs to make recommendations to a user based on the pages the user has already visited. In our preliminary experiments, we use our NCM to implement dynamic recommendations.

Note that there is a significant problem with user navigation patterns: a traveled path is not necessarily a desired path. Therefore we propose a recommendation mechanism which ignores those auxiliary pages and makes recommendations only on "relevant" pages. Of course the identification of "relevant" pages is a very difficult problem, but we can make assumptions based on certain heuristics. Though this definition is heuristic, we can still compare NCMs for how well they help users find the relevant pages more quickly.

The inputs to create an NCM include knowledge of page visiting patterns and the path of the user sessions; the outputs are recommendations for useful links. As mentioned above, a NCM can be created from the results of various learning methods, such as association rules, sequential patterns, and clusters. Moreover, a NCM can also be created with some pre-defined navigation templates.

### 5.1 Data Preparation

Our preliminary experiment uses only the server-produced access logs. Moreover, we experiment on two different original data sets so that we can make sure the result is not exceptional, and some comparisons can be done when necessary. Both data sets are the server access logs of our academic department: one is in the period of September and October, 2001 (*DS1*); another one is in the period of August and September, 2001 (*DS2*). In both data sets, we use the previous one-month logs as training set, and the later one-month logs as test set.

Our data preparation consists of breaking the access log into user sessions. Currently we use two different user identification methods: (a) we exclude all records without both authentication and cookie, and identify users just by their authentication or cookie value; (b) we identify users with all the information we have (including all log information as well as topology structure), and no record is excluded.

To compare these two user identification methods, we generated sessions with each of them on all the three-month logs. The result shows that with method (b), we can identify a lot more users and sessions. But interestingly, after we removed those useless sessions (with $length = 1$), the results from method (a) and (b) became quite similar (with a difference less than 2%). This implies that: (i) most users choose to accept cookies, not rejecting them; (ii) the large amount of "$length = 1$" sessions might come from some robots or web crawling programs.

In what follows, we use only method (a) for user identification, because it is more accurate and runs much faster than method (b). Another important parameter is the session timeout, for which we arbitrarily select 30 minutes.

### 5.2 Evaluation of Usage Improvement

Exercising our general framework, our experiment took three steps:

1. Convert original log data into sessions (data preparation).
2. Apply learning algorithms to the sessions to obtain NCMs. In our preliminary experiment, we only generate association-rule-based NCMs.
3. Apply the NCMs to the sessions in the test set, and generate a new set of shortened navigation paths.

The specific measure we use for evaluation is called *Usage Improvement (UI)*, which is computed based on the number of hyperlinks traversed:

$$UI = \frac{N_{org} - N_{com}}{N_{org}} \tag{1}$$

Here, $N_{org}$ is the number of requests in the original sessions, and $N_{com}$ is the number of requests in the compressed sessions.

For example, suppose we obtain an association rule $A \longrightarrow D$, where $D$ is a content page. Then an original session $S_{org} = \{A, B, C, D, E\}$ (where $B$ and $C$ are not content pages) can be compressed by removing $B$ and $C$ to obtain $S_{com} = \{A, D, E\}$. The usage improvement for this session would be $UI = \frac{5-3}{5} = 40\%$.

As mentioned above, our NCM mechanism only makes recommendations for relevant pages. Correspondingly, an association rule is applied only when it is used to shorten the path to a relevant content page. We determine relevant content pages using three different approaches: (a) *Maximal Forward Reference* (MFR) [3], which assumes that *maximal forward references are content pages*, and the pages leading up to the maximal forward references are auxiliary pages. Here, a maximal forward reference is defined to be the last page requested (before session timeout) by a user before backtracking; (b) *Reference Length* (RL) [3], which assumes that *a user generally spends more time on content pages than auxiliary pages*, therefore identifies content pages based on a cutoff viewing time. In our experiment, we set the cutoff time to an empirical value — 1 minute; and (c) *Visit Count* (VC), which simply assumes that those pages mostly visited are content pages.

Here we generated three kinds of Usage Improvement *UI*:

- $UI$ — usage improvement on all sessions.
- $UI_c$ — usage improvement on those sessions each of which has at least one content page.
- $UI_r$ — usage improvement on those sessions for each of which at least one rule is applied.

### 5.3 Experimental Results

We report two experiments. In the first, we tested the usage improvement with varying numbers of association rules and a fixed number of content pages. To simplify the problem, we set the minimal *confidence* of rules to a fixed value — 75%, and only adjusted the *support* of rules. The content pages were selected based on a restriction on their visit counts. So a web page is determined to be content page if: (a) it is classified to be a content page by one of the content page

identification methods (MFR-based, RL-based, or VC-based); and (b) it has a visit count no less than a threshold *vc*. In this experiment, we set *vc* to $\frac{S}{1000}$, where $S$ is the total amount of sessions. The number of content pages obtained with this approach may not be the same for different content page identification methods. In that case, we simply use the minimum among those numbers, and remove those less-visited pages when necessary.

In the second experiment, we tested the usage improvement with varying numbers of content pages and a fixed number of association rules. The support and confidence were set to 0.75% and 75% respectively. This is because smaller support and confidence values will generate a lot more rules without bringing apparent improvement to the performance.

The experiment results for *DS1* and *DS2* are shown exclusively in Figure 3.

First, we found that the total usage improvement ($UI$) we can expect is relatively small, though the usage improvement on those sessions where rules took effect ($UI_r$) is quite large. This means that only a very small part (less than 1% in our experiment) of the sessions were compressed with the association rules we obtained. A possible reason might be that, users' interest in this web site is too diverse, and the content pages we selected (1000 at most) are only a very small part of that. Moreover, among those three content page identification approaches, *reference length* seems to have the potential for the best Usage Improvement (*UI*).

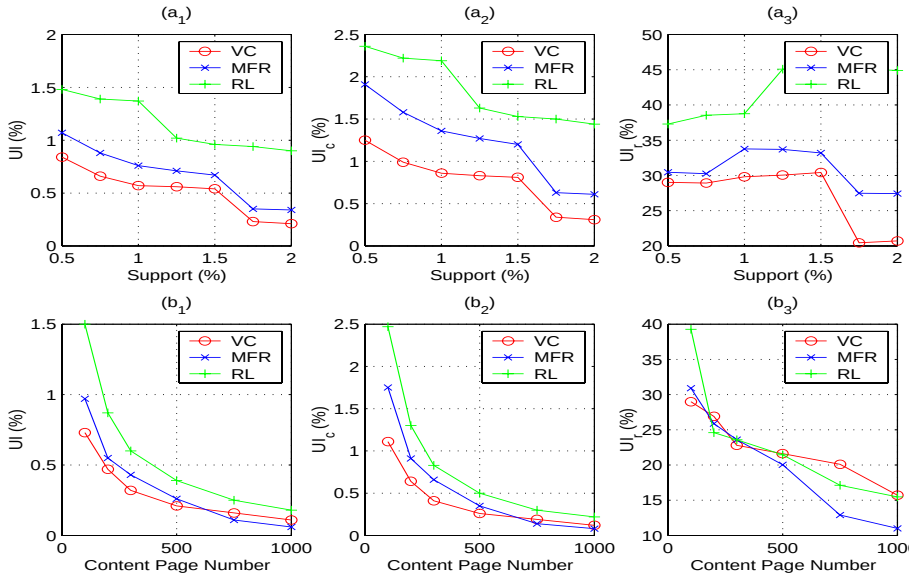
Secondly, with a given set of content pages, the usage improvement is improved when rule number gets larger (i.e., when the *support* gets smaller). However, this also means that users will have more recommendations to choose from in the real world, which can be seen as an extra cost.

Thirdly, we found that more content pages can have two effects: a positive one is that more paths can be compressed (no recommendation will be made if it is not for a content page, therefore no compression); however, there are more pages in the paths we can not skip. Our experiment showed that, with a given set of association rules, the usage improvement is impaired when content page number gets larger.

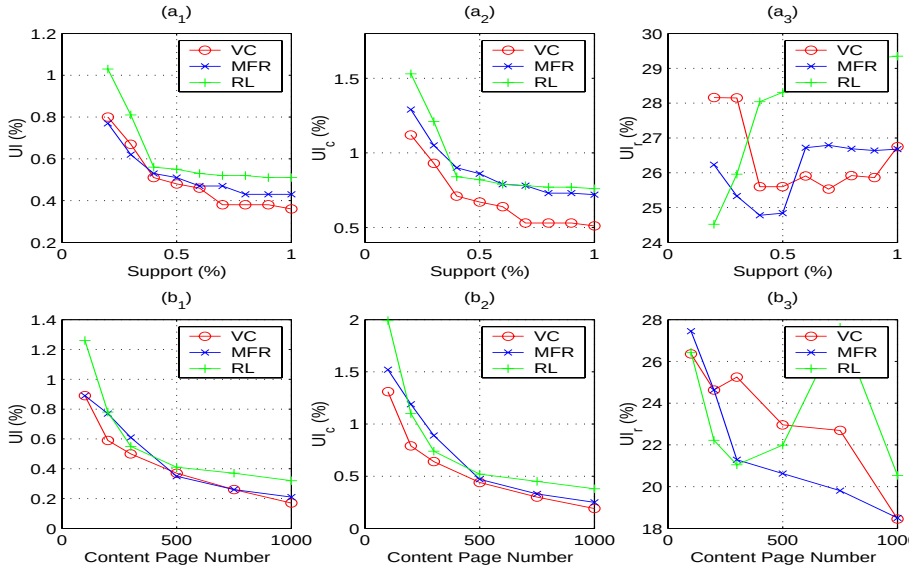
In those $UI = f(Support)$ figures, we can see that $UI$ can change a lot in some area, and changes very little in some other area. This means that not all rules have the same effect on $UI$: some rules are very useful, while some other rules can be completely useless. There might be three reasons behind this: (a) an association rule is possibly useful only when some content pages appear at its right side; (b) an association rule may not be useful even it has a content page at right side, because users might have already taken the path it suggests; (c) a lot of rules could relate to very few content pages.

### 5.4 Visualization of Navigation Compression Modules

Our visualization tool WV provides a way of comparing the navigation behavior of two different navigation strategies, by superimposing two sets of user navigation paths (in this case web logs). This is shown in Figure 4, where part (a) is the visualization of a user web log before applying a navigation learning method,



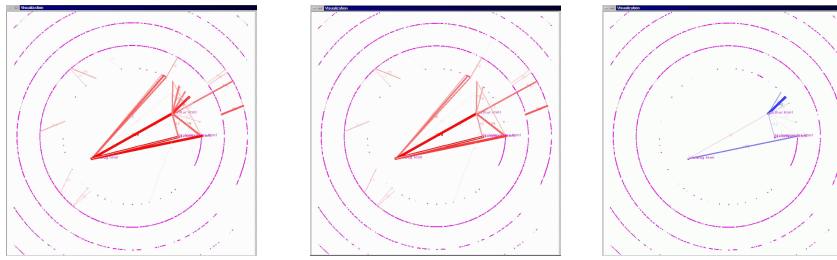

*(DS1)*




*(DS2)*

**Fig. 3.** Experiment Results on *DS1* & *DS2*

part (b) is a visualization of the same user web log *after* a learning method has been applied, and part (c) is a "subtraction" (b) - (a), which provides a visualization of the difference between the unimproved and improved navigation paths.



(a) before applying NCM (b) after applying NCM (c) pattern (b) - (a)

**Fig. 4.** Visualization of NCM application

## 6 Summary and Conclusions

We have developed a framework for web mining, based on a general architecture that decouples input data, learning method, evaluation method, and visualization. Our initial experiments with our framework focused on improving web navigation, and we developed the idea of a navigation compression model (NCM) to represent the results of learning "better" navigation paths from web logs.

In these our first experiments, our NCM's ability to make recommendations on "relevant" pages is important, but also revealed deeper problems. For example, we found that many pages were discarded in the compressed paths perhaps because we set the $vc$ too high. However, if we set the visit count threshold $vc$ too low, it means that few users' interest will become content pages for everyone. So there is obviously a trade off between aggregate and individual content page sets.

In future experiments we intend to try user-specific content pages for the evaluation. And in quantitative evaluation, the number of recommendations at each traversal step should be counted as an extra cost in the evaluation, because the more recommendations we have, the less possible that users will look into it. The determination of which pages to recommend is still a difficult problem.

We have initiated some experiments with other data preparation methods on different data sets but the NCM models differ only in size and so far have all been created with association rules. Our future experiments will create NCMs from other learning methods, and make comparisons among them.

## Acknowledgements

Our work is supported by the Canadian Natural Sciences and Engineering Research Council (NSERC), and by the Institute for Robotics and Intelligent Systems (IRIS) Networks of Centres of Excellence.

## References

[1] Igor Cadez, David Heckerman, and Christopher Meek. Visualization of navigation patterns on a web site using model_based clustering. In *Proceedings of KDD 2000*, August 2000.

[2] E.H. Chi, J. Pitkow, J. Mackinlay, P. pirolli, and J. Konstan. Visualizing the evolution of web ecologies. In *proceedings of CHI'98*, 1998.

[3] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems*, 1(1), 1999.

[4] Usama Fayyad, Georges G. Grinstein, and Andreas Wierse. *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann, 2001.

[5] Ed Huai hsin Chi and Stuart K. Card. Sensemaking of evolving web sites using visualization spreadsheets. In *proceedings of the Symposium on Information Visualization*, 1999.

[6] Paul Kahn and Krzysztof Lenk. *mapping web sites*. RotoVision SA, 2001.

[7] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. Technical Report TR99-010, Department of Computer Science, Depaul University, 1999.

[8] Tamara Munzner and Paul Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. In *VRML'95*, 1995.

[9] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[10] S. Schechter, M. Krishnan, and M.D. Smith. Using path profiles to predict http requests. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April 1998.

[11] J. Srivastava, R. Cooley, M. Deshpande, and P.N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. In *SIGKDD Explorations*, volume 1, Issue 2, January 2000.

[12] R. Stout. *Web Site Stats: Tracking Hits and Analyzing Traffic*. McGraw-Hill, 1997.

[13] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1982.

[14] I. Zukerman, D.W. Albrecht, and A.E. Nicholson. Predicting users' requests on the www. In *User Modeling: Proceedings of the Seventh International Conference (UM-99)*, pages 275–284, June 1999.