A Framework For Representing Navigational Patterns As Full Temporal Objects

AJUMOBI UDECHUKWU, KEN BARKER, AND REDA ALHAJJ
Advanced Database Systems and Applications Laboratory, Dept. of Computer Science, University of Calgary, *{ajumobiu, barker, alhajj}@cpsc.ucalgary.ca*

Navigational patterns have applications in several areas including: web personalization, recommendation, user-profiling and clustering, *etc*. Most existing works on navigational pattern-discovery give little consideration to the effects of time (or temporal trends) on navigational patterns. Some recent works have proposed frameworks for partial temporal representation of navigational patterns. This paper proposes a framework that models navigational patterns as full temporal objects that may be represented as time series. Such a representation allows a rich array of analysis techniques to be applied to the data. The proposed framework also enhances the understanding and interpretation of discovered patterns, and provides a rich environment for integrating the analysis of navigational patterns with data from the underlying organizational environments and other external factors. Such integrated analysis is very helpful in understanding navigational patterns (e.g., E-commerce sites may integrate the trend analysis of navigational patterns with other market data and economic indicators). To achieve full temporal representation, this paper proposes a navigational pattern-discovery technique that is not based on pre-defined thresholds. This is a shift from existing techniques that are driven by pre-defined thresholds that can only support partial temporal representation of navigational patterns.

Categories and Subject Descriptors: H.2.8 [**Information Systems**]: Database Management – *Database Applications*; K.4.4 [**Computing Milieux**]: Computers and Society – *Electronic Commerce*
General Terms: Design, Algorithms, Human Factors
Additional Key Words and Phrases: Navigational pattern discovery, web usage mining, temporal representation

1. INTRODUCTION

An interesting problem in Web usage mining [Srivastava et al. 2000] that has attracted the attention of several researchers is the discovery of traversal patterns (or link navigation patterns) of Web users [Chen et al. 1998]. Mining navigational patterns involves identifying how users access information of interest to them and travel from one object (e.g., web page) to another using the navigational facilities provided (e.g., hyperlinks). Tracking user-browsing habits provides useful information for service providers and businesses, and ultimately should help to improve the effectiveness of the service provided. Navigational patterns have several applications including the following: web recommendation systems and adaptive web sites [Niu et al. 2002], user profiling and web personalization [Mobasher et al. 2000], web page clustering [Smith and Ng 2003], web caching and pre-fetching [Yang et al. 2001], and several other applications. In this work, a broad definition of navigable systems is used, which includes all systems that provide information content to users, and also provide a means for the users to move around different sections of the system. The focus of this paper is the full temporal representation of discovered navigational patterns – i.e. representing navigational patterns as time series data. Such a representation would enable a full temporal analysis of the discovered patterns, and improve the understanding of the patterns.

This research was supported by the Natural Science and Engineering Research Council of Canada.
Authors' address: Department of Computer Science, The University of Calgary, Calgary, AB, Canada, T2N 1N4

Furthermore, traditional applications of navigational patterns (e.g., recommendation systems, user clustering, *etc*.) can be designed with increased robustness if the temporal trends and characteristics of the patterns are taken into consideration. Some recent works have addressed the problem of modeling patterns as temporal objects [Baron and Spiliopoulou 2001; Jin et al. 2002]. The techniques proposed in these works achieve partial temporal modeling of patterns (i.e., they do not represent patterns as full time series data with data points for each time unit). The major bottleneck in achieving full temporal modeling with these techniques is that they are based on pattern discovery algorithms that require pre-defined thresholds (e.g., support and confidence thresholds, or other distance metrics). This paper addresses this bottleneck by proposing a frequent path discovery algorithm that does not require pre-defined thresholds.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 presents the proposed framework for full temporal representation of navigational patterns and also discusses the proposed path mining technique that does not make use of pre-defined thresholds. Some motivating use cases for the proposed framework are presented in Section 4. Conclusions and areas of future work are discussed in Section 5.

2. RELATED WORK

Researchers in different fields have studied user navigational patterns extensively. Most of the research works on navigational pattern discovery, however, are in the area of web usage mining. Baron and Spiliopoulou [Baron and Spiliopoulou 2001] present a framework for monitoring temporal dynamics of discovered rules. Researchers have long acknowledged that rules change over time once the underlying data sources change. Thus, there are several works devoted to incremental approaches to knowledge discovery (see e.g. [Parthasarathy et al. 1999]). Incremental approaches generally address the problem of finding new rules that have emerged with the changes in the dataset, and removing previously discovered rules that may no longer be significant due to the changes in the data. Baron and Spiliopoulou [Baron and Spiliopoulou 2001] however argue that a rule may be significant at a point in time, disappear at a later point in time, and reappear much later. Thus, discovered rules may have inherent temporal characteristics. The paper however differs from the approach proposed in our research in several ways. First, the paper only studies the changes a rule may undergo when the underlying data changes, while in our research, inherent temporal trends in the data irrespective of data-updates are of interest. Second, the paper relies on existing rule discovery techniques for generating significant rules at each update. These techniques make use of support thresholds, thus all rules that meet the threshold are lumped together as existing at that timestamp while those that do not meet the threshold have no existence at the time-stamp in question. This approach has several limitations in temporal modeling. To overcome this shortcoming, our research proposes a frequent-path discovery method for preferred navigational paths that does not make use of support (or other pre-defined) thresholds, thus, all patterns can be represented as full, independent, temporal objects. A full temporal representation is necessary if the aim of applying conventional time-series analysis techniques to the discovered rules is to be achieved.

Baron et al. [Baron et al. 2003] extend the work done earlier [Baron and Spiliopoulou 2001] by proposing a technique for improving rule maintenance based on monitoring the temporal statistics of a subset of pre-discovered rules. The authors argue that changes in the base dataset are normally represented in the statistics of rules at any given timestamp.

Thus, instead of running computationally intensive data-mining algorithms on the entire dataset, the statistics of the pre-selected patterns are monitored. The underlying data-mining algorithm is only executed if there are significant changes in the statistics of the pre-selected rules being monitored. The paper may be seen as motivating an important application area for temporal modeling of rules but it is limited in its application and addresses a problem that differs significantly from the problem addressed in our research.

Jin et al. [Jin et al. 2002] explore the concept of distribution discovery in temporal rules. This work is similar to some of the concepts defined in [Baron and Spiliopoulou 2001]. The paper argues that local characteristics of temporal rules may have interesting global distributions. Thus, if the series is broken in time segments, there may exist distribution patterns in the resulting rules (i.e., a rule may be frequent, then infrequent, and frequent again, or a rule may only be frequent in the summer). The major distinguishing contribution of this paper is the development of several elegant techniques for identifying suitable time segments for breaking up the series. The notions of distribution in temporal rules used in this paper are very similar to some of the concepts explored in our research. However, Jin et al. [Jin et al. 2002] rely on specifying support thresholds (similar to [Baron and Spiliopoulou 2001]), so it does not support the total representation of rules as unique temporal objects, as explored in our proposed framework.

Li et al. [Lin et al. 2002] explore the correlation between rules and temporal segmentation (or calendar events), i.e., identifying patterns that cyclically occur at particular points in the calendar. The technique is apriori based, and requires the specification of the time segments of interest (i.e., the user has to specify if the rules should be assessed on a day-month-year interval, *etc*.). The goals of the paper are similar in spirit to some of the goals set out in our framework, in that our approach is also interested in identifying distribution and cyclical patterns in rules. However, the techniques proposed in our framework are not apriori-based, and do not require any knowledge of the underlying temporal segments (i.e., the user does not need to specify that interesting patterns will be found if a daily or weekly correlation is used, *etc*.).

Yang et al. [Yang et al. 2002] present techniques for extending sequential web-usage mining approaches to include a temporal dimension. The central question addressed in the paper can be summarized as follows: when (A, B, C) occurs, what is the next event that is likely to occur? Furthermore, if D is the most likely event to occur after (A, B, C), when will D occur? The authors use two sets of sliding windows to capture antecedent and subsequent events (these techniques are typically used to capture the reasonable lengths of a user's memory), thus the technique may be viewed as a hybrid n-gram approach [Su et al. 2000]. The focus of the paper, however, differs from the general temporal modeling of navigational patterns studied in our framework.

Several other authors have proposed algorithms for mining web usage patterns [Chen et al. 1998; Pei et al. 2000; Sarukkai 2000; Srivastava et al. 2000; Nanopoulos and Manolopoulos 2001; *etc*.] and sequential patterns [Srikant and Agrawal 1996]. These algorithms are mostly driven by support thresholds, and do not address issues related to temporal modeling of navigational patterns as discussed in our proposed framework.

3. METHODOLOGY

Baron and Spiliopoulou [Baron and Spiliopoulou 2001] present a framework for monitoring temporal dynamics of discovered rules. Unlike the techniques used in

incremental mining, the authors argue that a rule may be significant at a point in time, disappear at a later point in time, and reappear much later. Thus, discovered rules may have inherent temporal characteristics. We noted in Section 2 that the framework presented in [Baron and Spiliopoulou 2001] is limited in its ability to represent navigational patterns as full temporal objects. Our framework overcomes these limitations. The basic steps in our framework are as follows:

1. Select criteria of interest for generating the time series from the navigational patterns (for example, support count may be a criterion of interest).
2. Break-up the dataset based on some time segments. For example, given usage logs for one year, we can choose to place the logs for each day into a group. Several time-fragmentation strategies may be used. Multi-level analysis may also be employed if the time segmentation is organized into concept hierarchies (for example, day – month – year). Note that this is equivalent to the usual temporal referencing used in traditional time series analysis; for example, stock-market data may be collected and analyzed on a daily, monthly, or yearly basis.
3. Using a navigational pattern-discovery algorithm that does not require "apriori" or pre-defined thresholds (such as support thresholds), generate all the patterns in each time segment of the dataset, generating the criteria of interest specified in step 1 alongside. We use the term "apriori" here to refer to all thresholds that must be specified prior to (or at the early stages of) an algorithm run.
4. Represent each pattern discovered in step 3 as a time series $X = x_t$, $t = 1, \ldots, n$, where t is an index of time stamps, and n represents the number of time fragments specified in step 2. The entries in the time series (i.e., the x_t's) are the values for the criterion of interest.

The result is a time series for each pattern, for each criterion of interest. For example, given that the usage log is broken into monthly time segments and the log was collected over a one-year period. Also assume that "support-percentage" is the criterion of interest. Each segment of the usage log is processed and support-percentages are reported for each pattern. Assuming 'L', 'Q', and 'R' are objects (or webpages) in the system, and "LLR", "QQL", and "LL" are three patterns discovered from the usage-logs, the final representations of these patterns would be in the format given in Table I.

Table I. An Example Representation of Navigational Patterns as Time Series – using percentage support as criterion of interest

Pattern	**Time Segment**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
LLR	70	65	20	0	0	85	90	45	5	38	45	0
QQL	0	25	29	33	15	0	0	0	57	85	25	45
LL	88	80	75	70	77	80	81	85	83	85	80	88

In Table I, the time series for pattern "LLR" shows that it has 70% support in time segment 1, 65% support in segment 2, *etc*. It can also be seen that pattern "LLR" has a 0% support in segments 4, 5, and 12. This means that the pattern does not exist in those segments. Notice that this framework is based on the ability to discover patterns without using pre-defined thresholds. This means that all patterns existing in the usage logs are reported with their corresponding support counts (or other criterion of interest) for each time segment. Existing approaches for mining navigational patterns require the specification of some pre-defined threshold (usually a support threshold). Patterns that

meet the threshold are then output by the system, while those that do not meet the threshold are discarded. Such an approach is not suitable for our framework. To address this, we propose an algorithm that discovers navigational patterns without pre-defined thresholds. The proposed algorithm for discovering navigational paths of users is discussed in Section 3.1 below.

3.1 Discovering Navigational Paths of Users without Pre-defined Thresholds

The aim here is to find objects (or webpages) that are accessed together and the navigational patterns frequently exhibited by the users of the system. However, unlike the existing algorithms, the algorithm proposed in this section is not based on a predefined support (or other threshold). The technique presented here discovers contiguous (or consecutive) navigational patterns, a special case of constrained navigational patterns in which no gaps are allowed between discovered patterns. The proposed technique is as follows:
Step 1.) Define a unique symbol for each object/ web page in the system being analyzed.
Step 2.) Transform each user session into a string of literals using the defined symbols
Step 3.) Build a generalized suffix tree (GST) [Gusfield 1997] from the string literals obtained from transforming the user sessions. A suffix tree for a string *x* of length *n* is a rooted directed tree with exactly *n* leaves numbered 1 to *n*. The internal nodes of the tree, besides the root node, must have at least two descendants. The edges are labeled with nonempty sub-strings of *x* and no two edges originating from any particular node can have edge-labels that start with the same character. For any leaf *i* of the tree, the concatenation of the edge-labels on the path from the root to leaf *i* results in the suffix of string *x* from position *i*. Figure 1a shows the suffix tree for string "LLLR". A stop signal ($) is added to the end of the string to ensure explicit leaf nodes for all the suffixes of the string. The root node is depicted by a shaded oval, the internal nodes by unshaded ovals, and the leaf nodes by rectangles. Each leaf node is the path taken by a particular suffix of the sequence, and is named with the start position of that suffix. The label of each node is the concatenation of all the edge-labels for the path from the root to that node. To represent multiple strings on a suffix tree, a generalized suffix tree is built. For example, given that three user sessions have been transformed into the string literals "LLLR", "QQLLL", and "LL", a generalized suffix tree can be built as follows: append a stop symbol (e.g., '$') to all the strings; build a suffix tree for the first string (i.e., "LLLR" as shown in Figure 1a); traverse each of the remaining strings over the suffix tree in sequence, adding new nodes, edges, and leaves as needed. The leaf nodes in the generalized suffix tree may be shared by suffixes from different strings; thus, a list of the strings sharing the leaf nodes should be kept along with the corresponding suffix start positions. However, the suffix start positions are not necessary for the work discussed in this paper (because we are only interested in counts of unique strings sharing the nodes). Figure 1b shows an example of a generalized suffix tree built from the strings "LLLR", "QQLLL", and "LL". Notice that the labels of the internal nodes represent repetitions in the sequences.

In this work, the construction of the generalized suffix tree is adapted to capture repeated patterns in the transformed user sessions. The adaptations introduce two new properties for each internal and leaf node in the suffix tree. The first property is *string count*, which keeps a count of the number of strings (from the different strings being represented in the suffix tree) that have reached the internal or leaf node in question.

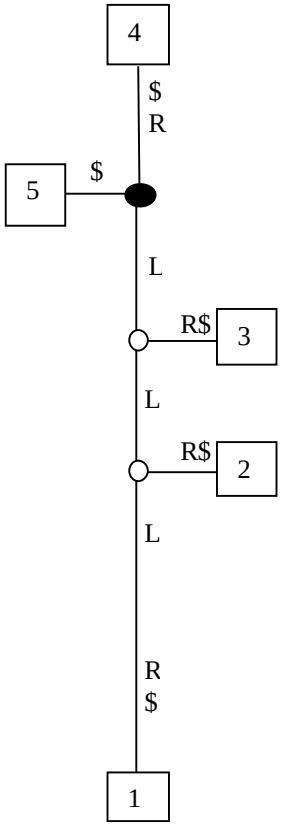

Figure 1a. Suffix tree for string "LLLR" with "$" appended as a stop symbol

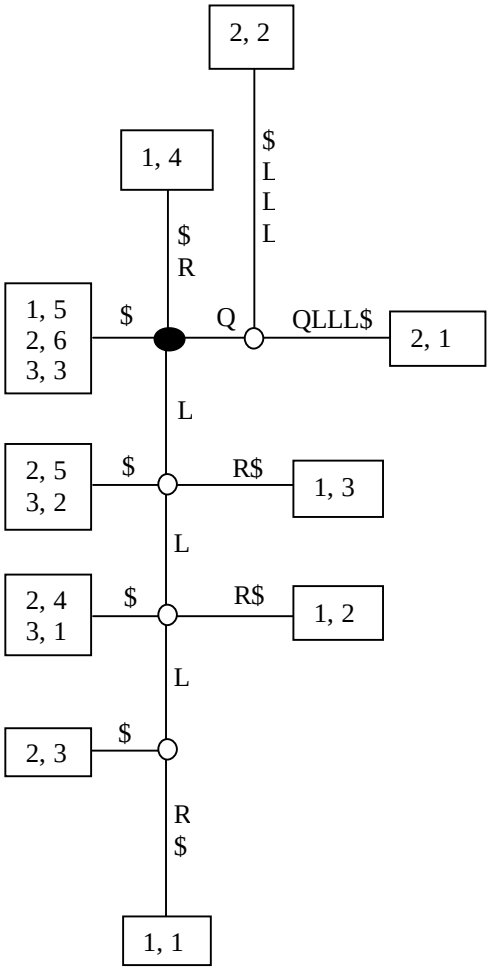

Figure 1b. An example of a generalized suffix tree for the strings "LLLR", "QQLLL", and "LL" with "$" appended to each string as a stop symbol

The second property is *last string*, which stores the identifier of the last string that reached that internal or leaf node. The *string count* and *last string* properties are also introduced for the edges in the suffix tree. The *last string* property of each edge in the suffix tree holds the identifier of the last string that traversed through that edge, while the *string count* property accumulates the number of unique strings whose suffixes have traversed through the edge.

When constructing the base suffix tree, i.e., the suffix tree for the first string; if a new internal or leaf node is created, *string count* is set to 1 for that node, and *last string* is set to the identifier of the first string (e.g., 1). For the subsequent strings added to the suffix tree after the base tree has been constructed, if a new internal node is created, the node's *string count* is set to 1 + *string count* of edge broken to create the node if the *last string* property of the edge being broken is different from the string-id of the current string being added to the tree; however, if the *last string* property of the edge being broken is

the same as the string-id of the current string being added to the tree, then *string count* of the new node is set to the *string count* of edge broken. The *last string* property of the new internal node is set to the identifier of the string being added. The *string count* property of a new leaf node is always set to 1. The *last string* property of a new leaf node is set to the identifier of the string being added. Each time a suffix of any string reaches an internal or leaf node, if the identifier of the string from which the suffix is taken is greater than the *last string* property of the node, the node's *string count* is incremented by 1 and the node's *last string* property is updated to the identifier of the string passing through (or reaching) that internal or leaf node. Similarly, each time a suffix of any string reaches the end of an edge, if the identifier of the string from which the suffix is taken is greater than the *last string* property of the edge, the edge's *string count* is incremented by 1 and the edge's *last string* property is updated to the identifier of the string passing through that edge. (This assumes that the identifiers of the strings are unique, and that the strings are included in the generalized suffix tree in increasing order of identifiers). Thus, the adapted generalized suffix tree stores all the repeated patterns in the transformed user sessions along with the count of user sessions that contain such patterns. For example, Figure 1b has 4 internal nodes (the unshaded ovals) with the following labels (recall that the label of a node is the concatenation of all the edge labels from the node to the root):

"Q"- traversed by string 2, i.e., *string count* = 1
"L"- traversed by strings 1, 2, 3, i.e., *string count* = 3
"LL"- traversed by strings 1, 2, 3, i.e., *string count* = 3
"LLL"- traversed by strings 1, 2, i.e., *string count* = 2

The example in Figure 1b is a case where all the repeated patterns are captured by internal nodes. In scenarios where two or more sessions are exact matches, the leaf nodes need to be accessed to capture all the repeated patterns. Thus, for a full enumeration of repeated patterns, both internal and leaf nodes need to be evaluated. For example, the four strings "LQR", "LQR", "LQ" and "QR" with "$" appended to each string as a stop symbol are represented in the generalized suffix tree shown in Figure 2. The String ID's are 1, 2, 3, 4, for the four respective strings. The leaf nodes (as depicted in the Figure) contain the *String ID's* and suffix positions of all the suffixes ending at the respective leaf nodes. The root node is the shaded oval, while the unshaded ovals are the internal nodes. Figure 2 has two internal nodes (the unshaded ovals) with the following labels (recall that the label of a node is the concatenation of all the edge labels from the node to the root):

"Q"- traversed by all 4 strings, i.e., *string count* = 4
"LQ"- traversed by strings 1, 2, 3, i.e., *string count* = 3

The rest of the repeated patterns can be retrieved by examining the leaf nodes. We are interested in the leaf nodes with adjoining leaf-edges that have labels other than the stop character ($), i.e., leaf nodes that contribute new patterns that have not been captured at the internal nodes. Figure 2 has six leaf nodes with the following labels (the leaf nodes are numbered in the Figure for ease of discussion):

Leaf 1: "LQR"- traversed by strings 1 and 2, i.e., *string count* = 2
Leaf 2: "QR"- traversed by strings 1, 2, 4, i.e., *string count* = 3
Leaf 3: "R"- traversed by strings 1, 2, 4, i.e., *string count* = 3
Leaf 4: adds no new pattern other than the stop symbol ($)
Leaf 5: adds no new pattern other than the stop symbol ($)
Leaf 6: adds no new pattern other than the stop symbol ($)

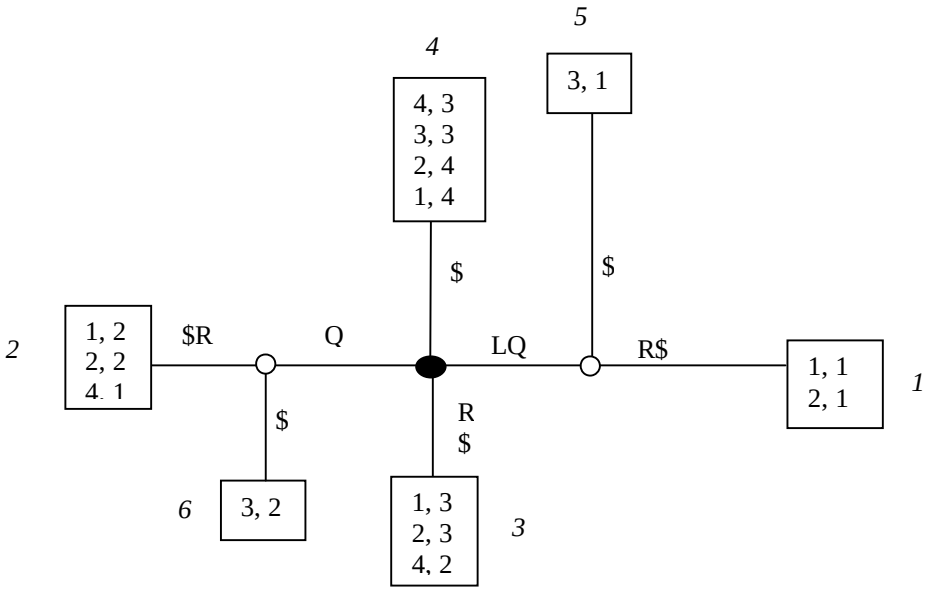

Figure 2. A generalized suffix tree with leaf-node relevance for the strings "LQR", "LQR", "LQ" and "QR" with "$" appended to each string as a stop symbol

Step 4.) At this stage in the process, all the navigational patterns are output to a database along with the count of user sessions in which they occur.

Step 5.) Patterns of interest may then be retrieved from the database of navigational patterns based on user-specified query parameters (e.g., support; that is the number of user sessions containing the pattern compared to the total number of user sessions).

We have implemented and tested the navigational path mining technique proposed in this section. We utilized publicly available datasets for our tests. All the tests reported here were run on a notebook PC, with 1GHz Intel celeron processor and 240MB RAM, running Windows XP home edition. The algorithms were implemented in Java, and run in the JBuilder 8 personal edition. Synthetic datasets were generated using the publicly available data generation program from the IBM Quest data-mining project[1], which has been used in several web-usage mining studies [Pei et al. 2000; Lu and Ezeife 2003]. The following parameters are used to generate the datasets: |D| - number of sequences in the database; |C| - average length of the sequences; |S| - average length of potentially frequent sequences; and |N| - number of objects in the database. We set |C| = 8; |S| = 4; |N| = 50; and vary |D| from 10,000 to 100,000 navigational sequences. We compare the performance of our technique to the apriori-based technique for discovering contiguous navigational patterns [CPY98]. (Note that the tree-based algorithms for discovering navigational patterns [Pei et al. 2000; Lu and Ezeife 2003] cannot differentiate contiguous navigational patterns, thus, we do not include them in our evaluations.) The results of our experiments are reported in Table II.

[1] http://www.almaden.ibm.com/cs/quest

Table II. Execution Times (in seconds) for Varying Database Sizes

	Number of Navigational Sequences									
	10K	**20K**	**30K**	**40K**	**50K**	**60K**	**70K**	**80K**	**90K**	**100K**
Adapted GST	2.2	4.1	6	8	9.9	11.6	13.8	16	17.7	19.6
Apriori-based (at 10% support)	10.1	20.4	30.2	40.3	50.8	60.5	71.9	82.2	90.1	101
Apriori-based (at 1% support)	50.1	100.7	156.1	207.2	265	297.9	366.4	440	467.1	514.3

We also tested the algorithms with publicly available, anonymous, real-world web logs of "msnbc.com" users [Hettich and Bay 1999]. We selected the first 100,000 navigational sequences and ran the adapted GST algorithm, and the apriori-based algorithm with 1% support threshold. The adapted GST algorithm ran in 29.3 seconds while the apriori-based algorithm ran in 68.8 seconds at 1% support threshold. Our results show that mining all contiguous navigational patterns utilizing the techniques discussed in this paper performs well compared to the existing techniques that make use of support thresholds.

4. DISCUSSIONS AND MOTIVATING EXAMPLES

So far in this paper, we have introduced techniques for representing navigational patterns as full temporal objects (or time series). In this Section, we present a few example scenarios where the proposed techniques would be useful.

Impact analysis of marketing campaigns: Let us examine the case of an E-commerce site. Generally, every E-commerce site would have some pages that are more popular than others (e.g., the home page). It would be an appropriate strategy to place marketing ads on the popular, more established pages. For example, an E-commerce travel site may have a well-established flight-booking business with loyal customers. Assuming the site also has a less popular hotel reservations business that it wants to promote. The company may periodically offer a sales promotion on the flight-reservation page offering discounts on hotel reservations. An important decision-support scenario here is to analyze the impact of the sales promotions. It is expected that the traffic to the hotel reservation page would increase with the sales promotion. It may be of interest to discover the trends in the navigation patterns before, during, and after the sales promotion. For instance, the business managers may wish to know if the sales promotions increase the number of loyal customers to the hotel reservation page. In cases where there are no permanent markets/customers created, it may be of interest to know how long the impact of the sales promotion was felt in the hotel-reservation site visits. This could be used in the timing of subsequent promotions. Several other scenarios can also be abstracted and analyzed.

Product/Page maturity analysis: In several websites (e.g., E-stores, University departments, *etc*.), several individual pages on the site may represent unique real-world concepts (e.g., products, people, *etc*.). When new pages are added to the site, they are often advertised in more popular pages (e.g., using banners, enhanced feature links, *etc*.). It may be of interest to study the usage trend to maturity of new pages – i.e., understanding the navigation to new pages from creation till a stable usage level. Such analysis would help in monitoring the advertisements on the popular pages to avoid cluttering.

Cluster migration/ interaction analysis: Navigational patterns have been successfully applied to web user profiling and clustering. An interesting extension to this application of navigational patterns may involve studying the formation and migration of user clusters. For example, it may be interesting to discover if the presence of particular clusters lead to the formation of some other cluster(s), i.e., studying the temporal causative relationships amongst user clusters. It may also be interesting to know if some clusters metamorphose into other clusters with time.

General temporal analysis: Several conventional temporal analysis techniques can also be applied to the time series generated from navigational patterns. Examples of such analysis include causative and correlation analysis – i.e., determining correlations between patterns and if the presence of certain patterns lead to the existence of other patterns. It may also be of interest to check for periodicities or cycles in pattern observations, or to discover emerging patterns, or surprising patterns, *etc*. All these analysis tasks would best be done in a wholesome framework, as they may be related to factors external to the system being studied (e.g., economic indicators, *etc*.).

5. CONCLUSIONS AND FUTURE WORK

Navigational patterns capture some useful behavioral patterns of clients of E-commerce sites. This paper advocates a paradigm shift in the way navigational patterns are discovered and represented. The paper proposes that navigational patterns can be represented and treated as full temporal objects (or time series data). To achieve full temporal representation, the paper proposes a navigational path-mining algorithm that does not make use of pre-defined thresholds. (Any pattern-mining algorithm that does not make use of pre-defined thresholds can be plugged into the framework proposed in this research). The proposed framework enhances the analysis, understanding, and interpretation of discovered navigational patterns. The framework also enables the integrated analysis of navigational patterns with external time series data (such as economic or market indicators). These benefits are easily transferable to applications based on navigational patterns (e.g., recommendation, personalization, user/ customer profiling, *etc*.), thus, improving E-commerce sites.

The path-mining algorithm presented in this work discovers consecutive navigational paths. An area of future work is to develop a path-mining algorithm for constrained navigational paths with gaps, that would not utilize pre-defined thresholds. It would also be interesting to build prototypes for some of the application areas discussed in Section 4.

REFERENCES

BARON, S., AND SPILIOPOULOU, M. 2001. Monitoring change in mining results. In *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery*, Munich Germany, 51-60.

BARON, S., SPILIOPOULOU, M., AND GÜNTHER, O. 2003. Efficient monitoring of patterns in data mining environments. In *Proceedings of the 7th East European Conference on Advances in Databases and Information Systems (ADBIS)*, Dresden, Germany.

CHEN, M-S., PARK, J.S., AND YU, P.S. 1998. Efficient data mining for path traversal patterns. *IEEE Transactions on Knowledge and Data Engineering, 10, 2*, 209-221.

GUSFIELD, D. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.

HETTICH, S., AND BAY, S.D. 1999. *The UCI KDD archive* [http://kdd.ics.uci.edu]. Department of Information and Computer Science, University of California, Irvine, CA.

JIN, X., LU, Y., AND SHI, C. 2002. Distribution discovery: local analysis of temporal rules. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, LNAI 2336*, Springer, 469-480.

LI, Y., NING, P., WANG, X.S., AND JAJODIA, S. 2002. Discovering calendar-based temporal association rules, *Data and Knowledge Engineering (DKE), 44, 2,* Elsevier, 193-218.

LU, Y., AND EZEIFE, C.I. 2003. Position coded pre-order linked WAP-tree for web log sequential pattern mining. In *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Seoul, Korea, 337-349.

MOBASHER, B., DAI, H., LUO, T., AND NAKAGAWA, M. 2000. Discovery and evaluation of aggregate usage profiles for web personalization. In *Proceedings of the WebKDD Workshop*.

NANOPOULOS, A., AND MANOLOPOULOS, Y. 2001. Mining patterns from graph traversals. *Data and Knowledge Engineering (DKE), 37, 3,* Elsevier, 243-266.

NIU, N., STROULIA, E., AND EL-RAMLY, M. 2002. Understanding web usage for dynamic web-site adaptation: a case study. In *Proceedings of the 4th International Workshop on Web Site Evolution*, Montreal, Canada, IEEE Computer Society Press, 53-62.

PARTHASARATHY, S., ZAKI, M.J., OGIHARA, M., AND DWARKADAS, S. 1999. Incremental and interactive sequence mining. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, ACM, 251 – 258.

PEI, J., HAN, J., MORTAZAVI-ASL, B., AND ZHU, H. 2000. Mining access patterns efficiently from web logs. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 396-407.

SARUKKAI, R.R. 2000. Link prediction and path analysis using Markov chains. In *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, The Netherlands, ACM Press, 377-386.

SMITH, K.A., AND NG, A. 2003. Web page clustering using a self-organizing map of user navigation patterns. *Decision Support Systems, 35, 2,* Elsevier, 245-256.

SRIKANT, R., AND AGRAWAL, R. 1996. Mining sequential patterns: generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT)*, 3-17.

SRIVASTAVA, J., COOLEY, R., DESHPANDE, M., AND TAN, P-N. 2000. Web usage mining: discovery and applications of usage patterns from web data, *SIGKDD Explorations, 1, 2,* ACM SIGKDD, 12-23.

SU, Z., YANG, Q., LU, Y., AND ZHANG, H. 2000. WhatNext: a prediction system for web requests using n-gram sequence models. In *Proceedings of the 1st International Conference on Web Information Systems Engineering*, Hong Kong.

YANG, Q., ZHANG, H.H., AND LI, T. 2001. Mining web logs for prediction models in WWW caching and prefetching. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, USA.

YANG, Q., WANG, H., AND ZHANG, W. 2002. Web-log mining for quantitative temporal-event prediction. *IEEE Computational Intelligence Bulletin, 1, 1,* IEEE, 10-18.