FAST EFFICIENT ASSOCIATION RULE MINING FROM WEB DATA

Omar H. Karam, Ahmad M. Hamad, Wedad H. Riad
Faculty of Computer and Information Sciences
Ain Shams University, 11566, Cairo, Egypt
ohkaram@asunet.shams.edu.eg, amhamad13@yahoo.com, wedad_hussein@hotmail.com

Abstract

The need for the analysis of the behavior of users on the World Wide Web motivated the use of data mining techniques for the discovery of traversal patterns. These patterns are usually expressed in the form of association rules. In this paper, we suggest a graph representation of the transactions database to assist with its division into a set of databases each containing fewer transactions and items. The runtime of the Apriori algorithm was compared when run on both the original and the divided databases. The division of the database was shown to improve the runtime by an average of 43.45% while maintaining the same results. Interestingness measures were also introduced as a way to improve the quality of the resulting rules. Introducing interestingness measures to the division process improved the average precision of the algorithm by a minimum of 15.5%.

Key Words

Web Usage Mining, Data Mining, Association Rules, Interestingness Measures.

1. Introduction

The need for the analysis of the users' click- stream data emerged from the reliance of many organizations on the World Wide Web to conduct business. As the number of users accessing a Web site grows the volume of click-stream data also grows. This fact motivated the use of data mining techniques for the analysis of this type of data.

One interesting type of patterns induced from click-stream data is association patterns, indicating which web pages are accessed together frequently. These patterns could be used for restructuring web sites, developing caching policies, or developing marketing promotions [1].

Generating association patterns from users' click-stream data stored in server logs goes through a set of steps. First, converting click-streams into sessions then, converting these sessions into transactions and finally, applying known association rule mining algorithms (such as the Apriori algorithm) on these transactions.

The basic problem with association rule mining algorithms is the runtime of such algorithms specially the frequent itemsets generation step. This step is very I/O intensive since it requires multiple passes on the transactions database for support count. Most of the contributions in the field of association rule mining [2, 3, 4] were targeted towards reducing the complexity of the frequent itemsets generation step. For example, the Apriori Algorithm [3] made use of the downward closure property of the support to reduce the number of candidate itemsets at each step of the algorithm. The downward closure property states that if an itemset is infrequent then all its supersets are also infrequent. In [4] a comparison of existing association rule mining techniques was presented.

Another problem with association rule mining algorithms is the quantity and quality of the resulting rules [5]. These algorithms produce a large number of rules a significant percentage of which is not interesting to the user. Interestingness measures [6] were introduced as a way to evaluate the quality of the resulting rules, and to restrict the results to rules that are interesting to the user.

In this paper we make use of the unique characteristics of web transactions to reduce the runtime of the frequent itemsets generation step of the Apriori algorithm when applied to web data. The contributions of our work include:

- Suggesting a graph representation of the transactions database to reflect the associations between pairs of web pages and the support of these associations;
- Using the previous graph representation to assist with the division of the transactions database into a set of smaller databases each containing fewer items in fewer transactions; and
- Introducing interestingness measures to the division process in an attempt to restrict the output of the algorithm to patterns interesting to the user.

The remainder of the paper is organized in the following way; in section 2, we present an overview of the association rule mining problem. An introduction to the characteristics of web transactions is given in section 3. The suggested graph representation is introduced in section 4. Our method for dividing the transactions database is described in section 5. In section 6 we introduce interestingness measures for further improvement. Performance results are presented in section 7. Section 8 contains the conclusions.

2. Association Rules

The problem of discovering association rules was first introduced in [2]. This problem could be stated as follows: let I be the set of all items in the database, given a set of transactions T where $\forall t \in T$, $t \subseteq I$, association rule mining algorithms seek to find rules of the form A→B (or, If A then B) where A and B are itemsets, $A, B \subset I$, and $A \cap B = \phi$. The metrics used to evaluate the significance of the resulting rules are support and confidence. Support is defined as the joint probability P(A,B), i.e. the percentage of transactions containing both A and B from the total number of transactions. While confidence is the conditional probability P(B|A), i.e. the percentage of transactions containing both A and B from the total number of transactions containing A. Generating association rules goes through two steps: First, finding frequent itemsets i.e. itemsets with support above a minimum support threshold, *minsupport,* and second, generating rules from these itemsets with confidence above *minconf,* where *minsupport* and *minconf* are parameters supplied by the user.

3. Web Association Patterns

Web server logs store the history of user requests and are the main data sources for web usage mining. Table 1 shows an example of typical server log entries which contains the host (user) accessing the page, time stamp, URL of the requested page, status, and size of each request. Entries in a web server log are sorted according to time stamp.

The first step for mining association rules from web data is generating transactions from log data. The transaction is usually expressed as an ordered sequence of web page references, while an item is a single web page. Maximal Forward References (MFRs) [7] have been proposed as a way to represent user access sequences. A Maximal Forward Reference is defined as the chain of references from the first page in a user's session until a backward reference is encountered. Dividing the sequence of references stored in a server log into sessions is required before identifying MFRs. A 30 minute timeout period has become a standard to perform this division [8]. The transaction database consists of the set of all MFRs generated from user sessions. Finally, known association rule mining algorithms would be applied to the transaction database to generate frequent itemsets. In our work we will use the Apriori algorithm for association rule mining and MFRs for transaction identification.

Time Window [9] is another method for transaction identification. It simply divides the log entries for a single user into time intervals not larger than a specified parameter. If the time window is sufficiently large, a transaction will contain all the references for a certain user.

The basic difference between web transactions and other types of transactions, like market basket data, is the significance of order. For market basket data the supports of the association patterns A→B and B→A are the same, which is not true for web data since visiting B from A is different than visiting A from B. Here we will use the definition of support used in [7]. The support of a web association pattern A→B will be defined as the percentage of transactions containing AB as consecutive references.

Another difference is in the way the candidate itemsets are generated. For association rule mining algorithms L_k denotes the set of frequent k-itemsets and C_k denotes the set of candidate k-itemsets. In the Apriori algorithm C_k could be generated by joining L_{k-1} with itself. Two itemsets $r_1, \ldots, r_{k-1}$ and $s_1, \ldots, s_{k-1}$ are joined to form a k-itemset if they have k-2 items in common. But because of the significance of order in web patterns the join process

Table 1. Example of a Web Server log

Host	Time Stamp	URL	Status	Size
199.72.81.55	[01/Jul/1995:00:00:01]	/history/apollo/	200	6245
pm1-5.america.net	[01/Jul/1995:01:01:33]	/images/NASA-logosmall.gif	200	786
winnie.fit.edu	[04/Jul/1995:00:02:21]	/history/apollo/apollo-10/apollo-10.html	200	3440

is modified. For any two itemsets $r_1, \ldots, r_{k-1}$ and $s_1, \ldots, s_{k-1}$ we join them to form a k-itemset only if either $r_1, \ldots, r_{k-1}$ contains $s_1, \ldots, s_{k-2}$ or $s_1, \ldots, s_{k-1}$ contains $r_1, \ldots, r_{k-2}$ [7].

4. Graph Representation of Web Transactions

In this section we suggest a graph representation of the transactions database to assist with its division. All the transactions in the database are represented in a single directed graph. The nodes in the transactions graph are the frequent web pages i.e. pages with support above *minsupport.* Arcs are represented as a triple (*source*, *destination*, *sup*), where *source* and *destination* are nodes in the transactions graph and *sup* is the support of the association pattern *source* → *destination.*

Constructing the transactions graph is combined with the generation of 2-itemsets and hence, it does not introduce any additional computational overhead to the process. Support pruning is next applied to the transactions graph to remove links with support below *minsupport* and divide the original graph into a set of sub-graphs. Figure 1 shows a simple illustration of the process. The shown graph is a representation of the following set of transactions {ACF, CF, ABDH, ABEI, DH, DGJ, DGJH, BEI}. The result of the previous process is a set of sub-graphs each representing a subset of the transactions database. Next, the subset of the transactions database corresponding to each sub-graph is identified and the Apriori algorithm is applied on each of these subsets individually.

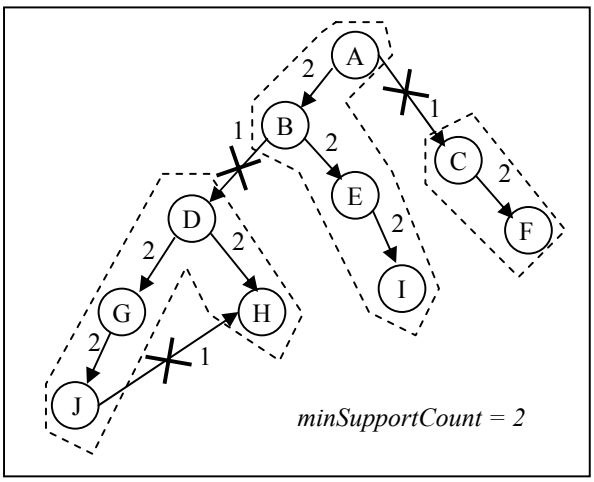

Fig. 1. Transactions Graph Division

It is important to note that the previous process is only applicable to web transactions. This is true because the users' behavior in such an environment is constrained with the structure of the site [1]. On the other hand for data such as market basket data, users are free to select any items together. Therefore, if we use our graph representation for market basket data the result would be an almost fully connected graph. Pruning such a graph will result in a single large sub-graph which will not reduce the overhead incurred in the frequent itemsets generation step. To the contrary, it will add to the process the overhead of executing the transactions graph division algorithm.

5. Dividing the Transactions Database

In this section we describe the details of the graph division algorithm, and its overheads.

5.1 Transactions Graph Division Algorithm

Let G be the set of links l (*source*, *destination*, *sup*) selected from the constructed graph after pruning, $subgraph_i$ is the sub-graph currently being processed, and T_i is the subset of the transactions database corresponding to $subgraph_i$. Figure 2 shows a formal description of the algorithm. A random link is selected from G and moved to the first sub-graph. Then all links connected to the current link are also moved to the same sub-graph. A link l_i is said to be connected to another link l_j if $source(l_i) = source(l_j)$ or $source(l_i) = destination(l_j)$ or $destination(l_i) = source(l_j)$ or $destination(l_i) = destination(l_j)$. The process is repeated for each new link added to the sub-graph, until there are no more links to add. Each link added to the sub-graph is removed from the original graph. The whole process is then repeated to construct the next sub-graph until there are no more links to be processed in G.

```
1. while G ≠ φ {
2.    subgraph_i = any link l in G
3.    T_i = transactions corresponding to l
4.    G = G – l
5.    for each link l in subgraph_i {
6.       L=all links in G connected to l
7.       subgraph_i = subgraph_i ∪ L
8.       T_i = T_i ∪ transactions corresponding to L
9.       G = G – L
10.   }
11.   Generate rules from Ti
12.   Move to next sub-graph
13. }
```

Fig. 2. Transactions Graph Division Algorithm

Note that moving a link to a sub-graph involves moving the subset of transactions corresponding to this link. The result of the previous process is a set of transaction databases each containing fewer transactions and items.

This reduces both the number of candidate itemsets and the number of transactions to consider when counting the support, and hence, reducing the runtime significantly.

5.2 Space Requirements

The Apriori algorithm stores candidate itemsets at each step of the algorithm in the main memory [3]. Our graph representation doesn't involve any additional storage requirements because the graph replaces the set of candidate 2-itemsets and have the same size. Also the need to store sub-graphs does not involve any additional requirements because as shown in the algorithm we only store one sub-graph at a time (step 9). Moreover, links added to any of the sub-graphs are removed from the original graph. Additional requirements come from the need to store the subset of the transactions database corresponding to the currently processed sub-graph.

5.3 Resulting Itemsets

Our transactions graph represents all possible paths existing in the transactions database. Given the method for generating candidate itemsets described in section 3, if a link was deleted from a certain path this path is broken and there could never be candidate itemsets generated containing items from both parts of the path. From the previous we reach the conclusion that applying the Apriori algorithm on the divided transactions database will produce the same results as when applied on the whole database.

6. Interestingness Measures

Data mining techniques usually produce a large number of patterns, but only a few of these patterns are of interest to the user analyzing the data. Interestingness measures are techniques required to reduce the number of patterns that need to be considered [10]. They increase the utility, relevance and usefulness of discovered patterns.

Interestingness measures are classified into two classes; *objective* and *subjective* measures. *Objective* measures depend on the structure of the pattern and underlying data, while *subjective* measures depend on the users' belief in the data [11].

After introducing database division as a method to enhance the runtime of the Apriori algorithm for association rule mining, we will use interestingness measures to improve the quality of the results of this algorithm. It is known that support and confidence are not enough to judge the usefulness or interestingness of association patterns [5]. This fact motivated the use of interestingness measures along with support and confidence to evaluate association rules. In our work we will use interestingness measures along with support as weights on the graph. This will cause the elimination of poorly correlated patterns early in the mining process. By applying the previous method we avoid the generation of candidate itemsets that will not yield interesting rules and hence, further reducing the number of candidate itemsets and improving the quality of the resulting patterns.

It was important to select the measures that best suite web data to be used as weights. In [12] a comparison of 21 interestingness measures for association patterns was introduced, and some properties were suggested that might be desirable for any interestingness measure. Each measure was tested against all these properties. Some of the suggested properties are required for our application, some contradict its requirements, and others are irrelevant to the application.

The selected measures are:

$$Addedvalue = P(B \mid A) - P(B)$$

$$Klosgen = \sqrt{P(A,B)}(P(B \mid A) - P(B))$$

These two measures were selected because they satisfy the following properties (for an association pattern A→B):

1. The value of the measure = 0 if A and B are statistically independent.
2. The value of the measure increases with P(A,B) when P(A) and P(B) are constant.
3. Asymmetry under Variable Permutations, i.e. they do not treat the following patterns equally [(A→B) and (B→A)].
4. Not Inversion Invariant: Inversion is like flipping "presence" to become "absence" and vise versa. Inversion Invariant measures treat the presence and absence of an item equally.
5. Not Null Invariant: Null Invariant measures produce the same result if we add more transactions that contain neither A nor B. Adding such transactions will reduce the support of the pattern.

7. Experimental Results and Discussion

Experiments described here were conducted on web server logs recorded from July 1st until July 7th 1995 at the NASA Kennedy Space Center WWW server in Florida [13]. The dataset contains a total of 542,509 requests. Generating transactions from this dataset resulted in 81,234 transactions. The distribution of the support count values for single web pages is shown in

Figure 3. The support count of and itemset is defined as the number of transactions containing the itemset. Web pages with very high support are usually navigational pages accessed only as means for exploring the web site and not for their content, or it could be the main page of the web site. Rules containing these pages are not interesting to the user because they contain information already known to him. Therefore, in our experiments we have pruned the candidate 1-itemsets using both minimum and maximum support thresholds to exclude such pages. A maximum support count of 1000 is used for all experiments.

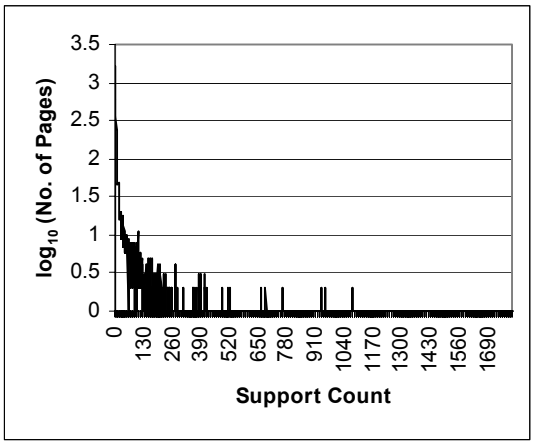

Fig. 3. Distribution of Support Count

7.1 Comparing Runtimes

In this experiment we studied the effect of graph division on the runtime of the frequent itemsets generation step. The experiments where conducted on a PC with a PIII 750 MHz processor and 256 MB of RAM.

The frequent itemset generation step was run twice; once on the whole database before division and another on the subsets of the transactions database corresponding to the set of sub-graphs. The runtimes in both cases were recorded.

The generation of the 1-itemsets and 2-itemsets is a common process in both cases. Therefore, the runtime of this process is added to all the previously recorded runtimes. Figure 4 shows the recorded runtimes for different values of *minSupportCount*. The improvement in the runtime is shown on the chart for each value. From the results we find that dividing the transactions database improves the runtime by an average of 43.45%.

It is also to be noted from the chart that the improvement in the runtime decreases as the value of *minSupportCount* increases. This is true because as the value of *minSupportCount* increases the size of the pruned graph decreases and approaches the size of individual sub-graphs. This happens while the time required to perform the graph division almost remains the same.

In section 5.2 it was shown that the storage overhead in our algorithm comes from the need to store the subset of the transaction database currently processed by the mining algorithm. Experiments have shown that the size of this subset does not exceed 5% of the size of the original transactions database.

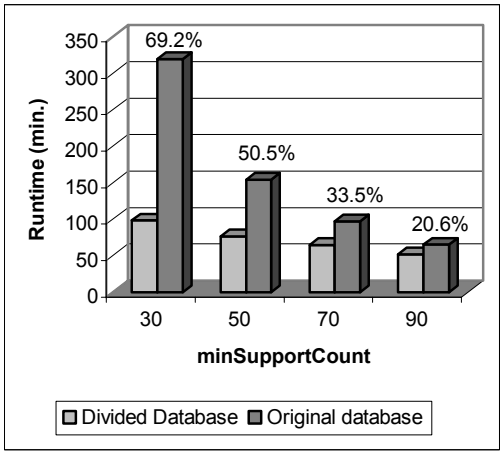

Fig. 4. Comparison of Runtimes

7.2 Using Interestingness Measures

In this experiment we studied the effect of using interestingness measures as weights on the graph. We use Precision as a metric to evaluate the quality of the results. Precision is defined as the percentage of interesting rules from the total number of rules retrieved. We will compare Precision values when using support only as weight on the graph or when using it along with an interestingness measure. Precision is compared for different values of *minSupportCount* and for different thresholds on the value of the interestingness measures. The threshold is chosen as a percentage of the range of values of the interestingness measure associated with the links in the graph. The experiments are repeated for the two measures chosen before (Added Value and Klosgen's). The threshold is determined at the graph division step. The same threshold value used to prune the graph is used to identify interesting patterns form the results. Figure 5 shows the results of the experiments. As could be seen from the results the precision values when using interestingness measures as weights on the graph along with support are always higher than those recorded

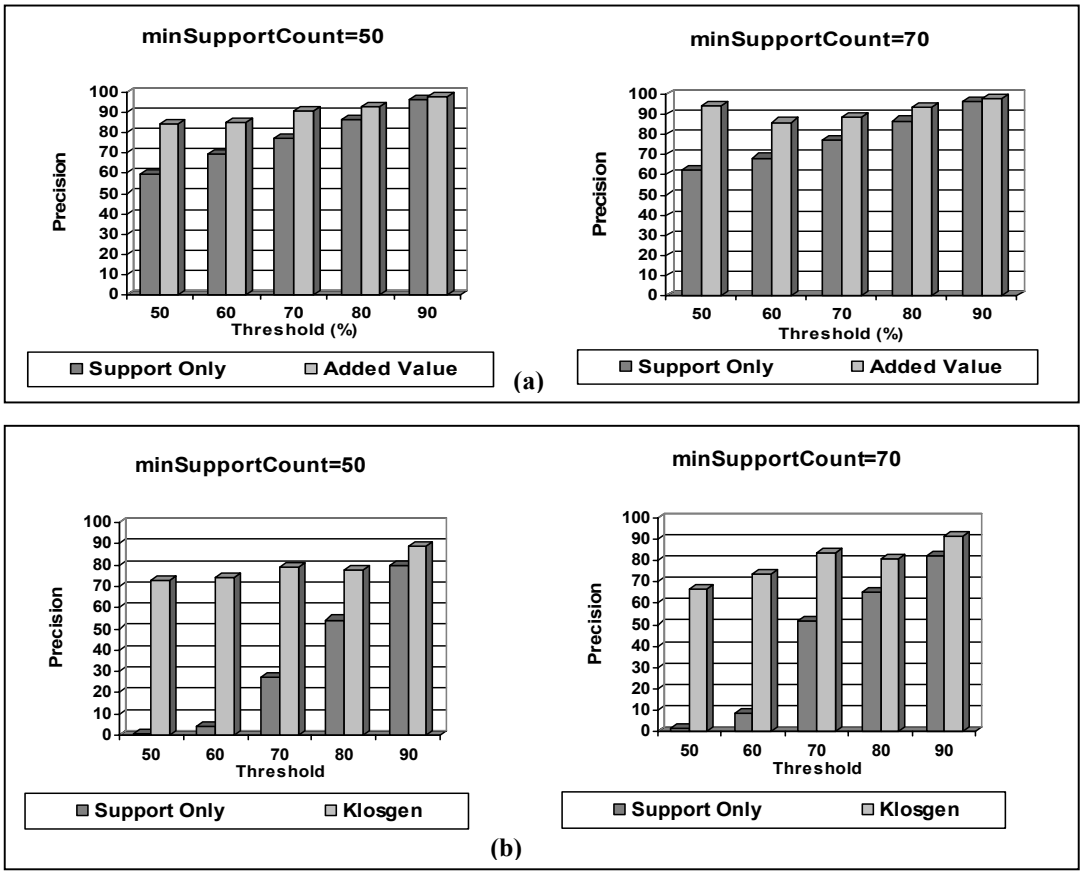

Fig. 5. Comparison of Precision Values (a) Using Added Value as Weight (b) Using Klosgen as Weight

when using support only. That's because using interestingness measures as weights eliminates non-interesting patterns early in the mining process. Table 2 shows the improvements in the average precision of the algorithm for all cases.

Table2. Improvements in Average Precision

Interestingness Measure	minSupport Count	Improvement in Average Precision(%)
Added Value	50	15.5
	70	17.5
Klosgen	50	135
	70	89.7

It is worth noting that the values given in Table 2 do not mean that Klosgen is a better measure than Added Value, in fact the precision values for both measures are close. But what really happens is that when using Klosgen to judge the interestingness of the rules resulting from the mining algorithm while using support only as a weight, the precision values drop significantly.

Note that $Klosgen = \sqrt{P(A,B)}(AddedValue)$. The values of P(A,B) for the dataset used in this paper are very small since the support count values does not exceed 1700 while the transactions database contains 81,234 transactions. Because of this fact the value of P(A,B) has the dominating effect on the value of Klosgen, and hence the distribution of the values of the measure is very similar to the distribution of support count values shown in Figure 3. From the chart we could see that most pages are located in the lower half of the support range, and the same applies for Klosgen. From the previous we could reach the conclusion that some patterns that are considered interesting by Added Value may be non-interesting according to Klosgen, and hence the precision values are lower when using Klosgen to judge the interestingness of patterns.

8. Conclusion

In this paper, a graph representation of web transactions was suggested to assist with the association rule mining process. An algorithm was introduced to divide the previous graph into a set of sub-graphs and consequently divide the transaction database into a set of smaller databases using minimum support pruning. The mining algorithm was run on the original and the divided database for 4 different values of *minSupportCount*. The runtimes were compared for all cases. The experiments showed an average improvement of 43.45% in runtime when using the divided database.

Also, interestingness measures were introduced as weights on the graph to both assist with the graph division and to eliminate non-interesting patterns early in the mining process. Two measures were selected for this task; Added Value and Klosgen's. Precision values were compared for different values of *minSupportCount* and different thresholds on the interestingness measures. Experiments showed a minimum improvement for the average precision values of 15.5%.

9. Acknowledgement

The logs used in this work were collected by Jim Dumoulin of the Kennedy Space Center, and contributed by Martin Arlitt and Carey Williamson of the University of Saskatchewan.

References

[1] P. Tan, and V. Kumar, Mining association patterns in web usage data, *International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*, 2002.

[2] R. Agrawal, T. Imielinski, and A. Swami, Mining association rules between sets of items in large databases, In *proceedings of ACM SEGMOD International Conference on the Management of Data (SIGMOD'93)*, Washington, D.C., May 1993, 207-216.

[3] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, In *Proceedings of the 20th International Conference on Very Large Databases (VLDB)*, Santiago, Chile, September 1994, 487-499.

[4] J. Hipp, U. Guntzer, and G. Nakaeizadeh, Algorithms for association rule mining - A general survey and comparison, In *proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2000, 58-64.

[5] P. Tan and V. Kumar, Interestingness measures for association patterns: A perspective, *KDD 2000 Workshop on Postprocessing in Machine Learning and Data Mining*, 2000.

[6] G. Piatetsky-Shapiro, Discovery, analysis and presentation of strong rules, In *Knowledge Discovery in Database,* AAAI/MIT Press, 1991, 229-248.

[7] M.S. Chen, J.S. Park, and P.S. Yu, Data mining for path traversal patterns in a web environment, In *Proceedings of the 16th International Conference on Distributed Computing Systems*, 1996, 385-392.

[8] J. Srivastava, R. Cooley, M. Deshpande and P. Tan, Web usage mining: Discovery and application of usage patterns from web data, *SIGKDD Explorations, 1*(2), 2000, 12-23.

[9] R. Cooley, B. Mobasher, and J. Srivastava, Grouping web page references into transactions for mining world wide web browsing patterns, *Technical Report TR 97-021*, University of Minnesota, Dept. of Computer Science, Minneapolis, June 1997.

[10] R.J. Hilderman and H.J. Hamilton, Knowledge discovery and interestingness measures: A survey, *Technical Report CS 99-04*, Department of Computer Science, University of Regina, October 1999.

[11] A. Silberschatz and A. Tuzhilin, What makes patterns interesting in knowledge discovery systems, *IEEE Transactions on Knowledge and Data Engineering*, *8(6)*, December 1996, 970-974.

[12] P. Tan, V. Kumar, and J. Srivastava, Selecting the right interestingness measure for association patterns, In *proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD-2002).

[13] http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html