

Mining the Most Interesting Web Access Associations

Li Shen, Ling Cheng, James Ford,
Fillia Makedon, Vasileios Megalooikonomou, Tilmann Steinberg
The Dartmouth Experimental Visualization Laboratory (DEVLAB)
Department of Computer Science, Dartmouth College, Hanover NH 03755
{li, ling, jford, makedon, vasilis, tilmann}@cs.dartmouth.edu

Abstract: Web access patterns can provide valuable information for website designers in making website-based communication more efficient. To extract interesting or useful web access patterns, we use data mining techniques which analyze historical web access logs. In this paper, we present an efficient approach to mine the most interesting web access associations, where the word "interesting" denotes patterns that are supported by a high fraction of access activities with strong confidence. Our approach consists of three steps: 1) transform raw web logs to a relational table; 2) convert the relational table to a collection of access transactions; 3) mine the transaction collection to extract associations and rules. In both step 1 and step 2, we provide users with an effective mechanism to help them generate only "interesting" access records and transactions for mining. In the third step, we present a new efficient data mining algorithm to find the most interesting web access associations. We evaluate this approach using both synthetic data sets and real web logs and show the efficacy, efficiency and good scalability of the proposed mining methods.

Introduction

The World Wide Web is rapidly emerging as an important communication means for the dissemination of information related to a wide range of topics (e.g., education, business, government, recreation). In order to organize a website well and provide the most attractive and valuable information to users, the designer is usually very interested in understanding the access behavior and patterns of the users. Data mining techniques (Agrawal et al. 96, Garofalakis et al. 99, Kwedlo et al. 98, Mobasher et al. 96, Shafer et al. 96, Shen et al. 98) can aid in the extraction of these useful access patterns by analyzing historical web access logs.

We are developing a web log mining system, which aims to discover useful web access patterns like association (Agrawal et al. 93) and classification rules (Shafer et al. 96) based on users' interests. Figure 1 shows the overall structure of our current system. The goals of this research are twofold. On one hand, we design new efficient data mining algorithms and use web logs as real data sets to test their performances. On the other hand, we study how to apply our algorithms or adapt existing data mining techniques to web access pattern mining. Our system currently consists of three modules: 1) association miner: finds correlations between pages that are often accessed together; 2) decision tree classifier: creates a decision tree to do classification (e.g., classifying webpages based on access attributes like domain type, time and method of the request) (Shafer et al. 96); 3) evolutionary approach (EA) classifier: uses EA (Kwedlo et al. 98) instead of decision tree based approach to generate classification rules. In this paper, we focus on the first module and present our approach on mining the most interesting web access associations, where the word "interesting" denotes patterns that are supported by a high fraction of access activities with strong confidence.

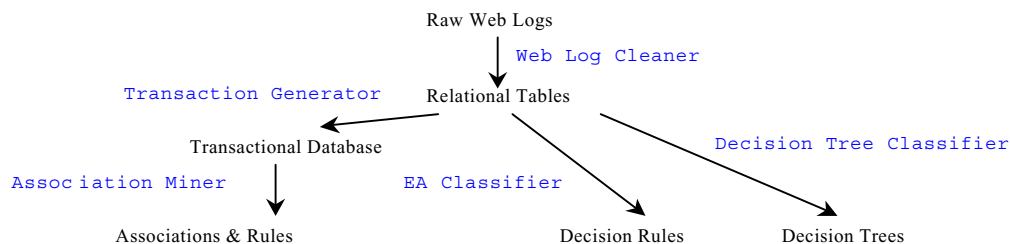


Figure 1: Web access pattern mining system structure.

Association rules, introduced in (Agrawal et al. 93), provide a useful mechanism for discovering correlations among items belonging to customer transactions in a market basket database. An association rule has the form $X \Rightarrow Y$, where X and Y are sets of items or itemsets. Let the support of an itemset X be the fraction of database transactions that contain X . The support of a rule of the form $X \Rightarrow Y$ is defined as the support of $X \cup Y$, while its confidence is the ratio of the supports of $X \cup Y$ and X . The association rules problem is that of computing all association rules that satisfy user-specified minimum support and minimum confidence constraints.

In the web log mining context, we are interested in finding associations like "x% of the people who visit page A also visit page B". Generally, this type of patterns can give some guidance to website designers about what pages should be linked or what contents should be placed together. Discovery of such rules for organizations engaged in E-Commerce can help in the development of effective marketing strategies. This can also help in educational systems. Many professors have been using the Web as an important means to make announcements, provide materials, and give assignments to their students. It is often interesting for them to know how students visit these course websites and what pages are typically accessed together, this kind of information can help them better organize their classes.

In this paper, we present a new efficient association mining algorithm and apply it to the problem of web access association mining. Our approach can be divided into 3 steps: 1) web log cleaner: transforms raw web access log to a relational table; 2) transaction generator: converts the relational table to a collection of user access transactions; 3) association miner: mines the most interesting associations from the collection of web access transactions. The first two steps are mainly data preparation procedures, which are similar to those presented in some other papers such as in (Mobasher et al. 96). However, in these two steps, we also introduce some practical means to help users prepare data based on their interests so that only relevant access records and transactions are considered in the association mining procedure. The third step is accomplished by a new efficient association mining algorithm MFA (Most Frequent Associations). MFA is an extension of our previous work (Shen et al. 98) and the famous Apriori (Agrawal et al. 96) algorithm, which is used by most of current web access association mining systems (e.g., Garofalakis et al. 1999, Mobasher et al. 1996). By avoiding an exponential running time bottleneck as well as using the minimum support constraint, MFA has a better performance than (Shen et al. 98) and Apriori.

The rest of the paper is organized as follows. We describe our web log cleaner in Section 2, transaction generator in Section 3, and association mining algorithm in Section 4. We do a performance study in Section 5 and conclude the paper in Section 6.

Web Log Cleaner

Since the format of the collected web log data is not suited for direct import into the mining algorithms, it is important to transform and clean this data. The web mining system we are developing includes components for finding classification and association patterns. Because of this we first transform the raw data to a relational table, a suitable format for doing classification. Our transaction generator will later convert the relational table to a collection of access transactions for mining associations.

To convert the raw web logs into a relational table, we introduce a schema file. The function of the schema file is to tell the web log cleaner both the formats of the raw data and of our target relational table. The schema file can be defined flexibly to meet the different requirements in different situations. For example, web servers may have different log formats, and users may have different interests in access record attributes.

In our experiments, we use as test data the web logs in the computer science department at Dartmouth. Our access logs are generated by the logging module of Apache 1.3.6, which is an extension of Common Log Format. Please refer to http://www.apache.org/docs/mod/mod_log_config.html for detailed log format information. We extract the following attributes to form our relational table: Host, Day, Month, Year, Time, Method, Location, Bytes; where Host is the domain name or IP address of the request, Day/Month/Year/Time is the time stamp of the request, Method is the method of request (GET or POST), Location is the name of the file requested, and Bytes is the size of the data sent back.

In our schema file, we also provide effective mechanisms for users to generate a customized relational table based on their interests. These mechanisms work only for categorical attributes (coming from an unordered domain), not for numeric attributes (coming from an ordered domain). Given a categorical attribute, users can define a set of strings of the following two types: 1) significant string x : the relational table includes only records with the corresponding attribute value that contains x . 2) negligible string y : the relational table includes only records with the corresponding attribute value that does not contain y . For attribute "Location", users can define several group strings, say "*.z", such that all files with name matching "*.z" will be treated as one group value only.

Thus, using significant string ".edu" for Host, users can generate all access records which are coming from educational institutions. Using negligible string ".gif" for Location, users can exclude all access records targeting any GIF file. This is very useful, since users usually want to filter out unwanted entries like accesses to image files that were embedded in a web page whose 'hit' had already been logged. Also, using (for example) group string ".pdf" for Location, users can group all PDF files to a single group value. This restriction is especially useful for association mining, since downloading lots of PDF files often form large transactions, which is not only unnecessary but also bad for the performance of association mining algorithms. In short, these restrictions can help users focus on their interests and improve the quality of mining results, and can also help improve the performance of the mining algorithms.

Transaction Generator

To apply association mining algorithms to web log mining, one needs to work on a collection of transactions. Unlike market basket analysis, where a single transaction is defined naturally, we do not have a natural definition of web association transactions. Since we are interested in finding pages that are often visited together, a transaction should consist of a set of Locations (i.e., requested files). We define a transaction based on the set of all log entries belonging to the same host (domain name or IP address), within a given time interval (provided by the user; we use 1 hour in our experiments) (Mobasher et al. 96). Our transaction generator scans over the relational table obtained in the step 1 and generates a collection of access transactions, each of which is a group of locations belonging to the same host within a given maximum time gap.

To help users focus on interesting access transactions, we provide them with two parameters: minTranSize and maxTranSize. A transaction with size between minTranSize and maxTranSize is considered to be a transaction of good size. The output of our transaction generator includes only transactions of good size. By setting, for example, minTranSize to 2, users can focus on all transactions containing at least two locations, since associations can only happen between at least two locations and some users may regard all singleton transactions as uninteresting. Another example of using the parameters can be found in an E-Commerce application. In an on-line shop website scenario, it is natural for the analyzer to distinguish between serious access transactions and occasional tourist access transactions based on the number of hits included in the access session. It is also valuable to find the different access associations between these two types of access transactions. Our mechanism can help approach this goal, if, for example, we choose to view serious transactions as those with size above 10 and occasional tourist transactions as those with size below 10.

Mining the Most Frequent Itemsets

After finishing the data preparation for a collection of access transactions, we focus on our association mining task. As we mentioned before, an association rule has the form $X \Rightarrow Y$, where X and Y are sets of items or itemsets. The association rules problem is that of computing all association rules that satisfy user-specified minimum support (MinSup) and minimum confidence (MinConf) constraints. The problem can be divided into two subproblems: 1) finding all frequent itemsets (that is, itemsets that satisfy MinSup); and 2) finding all strong rules (that is, rules that satisfy MinConf) from all frequent itemsets. The second subproblem is relatively straightforward. The following is a naïve but feasible algorithm: for each frequent itemset a and each subset b of a , if $\text{support}(a)/\text{support}(b) \geq \text{MinConf}$, then output $b \Rightarrow (a - b)$. Almost all previous studies focus on the first subproblem: computing frequent itemsets. Among them, the Apriori algorithm (Agrawal et al. 96) is the most popular one, which is why we do experimental comparison between our approach and Apriori in Section 5.

Based on the above problem definition, we observe that an inappropriate MinSup may imply an exponential number of frequent itemsets and cause any solution to have an exponential running time. In practice, users sometimes have no easy control on setting MinSup and may run into this exponential bottleneck. Note that it is much easier to say "top 100" and "top 1000" than set MinSup. Based on this, we introduce an interesting new problem by adding a new result size constraint N : the desired number of frequent itemsets. In (Shen et al. 98), we have presented an algorithm for mining the N most frequent itemsets without considering MinSup. The algorithm has good theoretical performance, i.e., running in polynomial time (in terms of N) for practical applications. However, due to lack of MinSup constraint, it may run more slowly than Apriori in practice when N is large and

MinSup is high. Also, if N is set too small, most of the frequent itemsets generated are 1-itemsets, from which interesting association rules cannot be derived.

We have derived an efficient algorithm, MFA (Most Frequent Associations), to remove the above disadvantages by mining the top N most frequent itemsets (with size>1) satisfying MinSup constraint. Its advantages are: 1) as an extension to (Shen et al. 98), theoretically it runs in polynomial time for practical applications (see (Shen et al. 98) for a proof); 2) using both N and MinSup as constraints, it has better performance than Apriori and (Shen et al. 98); 3) it uses N constraint only for counting the number of frequent itemsets with size>1 so that enough association rules can usually be guaranteed to be obtained.

MFA computes frequent itemsets in passes. At pass k, MFA finds the N most frequent itemsets with size >1 and satisfying MinSup (called k-winners) among U_k , the set of all itemsets of size $\leq k$. MFA uses W_k to store all k-winners, and uses k-CriSup to denote the support of the most infrequent k-winner. W_k is used to generate C_{k+1} , the candidate itemsets for whom support is counted in the (k+1)-th pass. The key idea is that an itemset can be pruned from C_{k+1} if any of its k-subsets does not belong to W_k because of the following reason: we have $k\text{-CriSup} \leq (k+1)\text{-CriSup}$, since the candidates competing for (k+1)-winners form a superset of the candidates competing for k-winners; thus, given any (k+1)-itemset, say x, it satisfies (k+1)-CriSup and so satisfies also k-CriSup, which implies that any k-subset of x must be a k-winner. MFA terminates either when k reaches the total number of items or when C_k is empty.

Performance Study

We have implemented our web association mining approach in our web mining system and done an experimental study on both real web access logs and synthetic data sets. We present some of our experimental results here. Our system is written by C and the experiments on real web access logs were done on an SGI with 128 MB memory running IRIX 6.5. The experiments on synthetic data were done on a DEC Alpha 500/333 with 512 MB memory running Digital UNIX V4.0F. We measure performance by looking at, (1) for real web logs, if our approach can find interesting results within reasonable amount of time, and (2) for synthetic data, if our approach runs faster than previous algorithms and has good scalability.

Tests on Real Web Logs

In this test, we try to find if our approach can efficiently find interesting patterns from a large number of real web access log entries. We have collected 3 months (from 9/12/99 to 12/12/99, roughly the Dartmouth College's 1999 Fall Term) web access logs from our department's web server, consisting of 3559272 entries. In the experiments, we use negligible strings ".gif", ".jpg", and ".xbm" to ignore these image files, and group strings ".pdf", ".ps" and ".bib" to group related files to the corresponding group values. We always set MinSup count (referring to the number of transactions) to 20, result set size N to 200, and MinConf to 50%.

We use minTranSize.maxTranSize.range to denote the mining task we performed, where range can be one of {all, cs5, cs88}, 'all' denotes no significant string constraint, 'cs5' and 'cs88' denote using "~cs5" and "~cs88" (referring to two Dartmouth class accounts) as significant string constraints, respectively. For example, '2.30.cs5' denotes the mining task which focuses on all access transactions with size between 2 and 30, where the involved access records are all from accesses to cs5 class website. Table 1 summaries test results on 6 mining tasks, where we uses these parameters - goodTnum: the number of transactions in good size; prepTime: the total running time of step 1 and step 2; miningTime: the running time of step 3; AssoN: the number of final winners (i.e., frequent itemsets) with size>1; RuleN: the number of strong rules extracted from frequent itemsets.

Task	goodTnum	prepTime	miningTime	AssoN	RuleN
1.100.all	419384	1431.04	357.91	200	488
2.30.all	182753	1437.5	518.08	200	587
1.100.cs5	12135	518.93	2.87	200	327
2.30.cs5	10632	521.72	2.15	200	333
1.100.cs88	774	496.76	0.13	67	67
2.30.cs88	591	495.07	0.12	64	78

Table 1: Test results on 6 mining tasks.

From Table 1 we see that all these mining tasks produce a suitable number of interesting patterns within reasonable amounts of time. We also find that the good transaction size setting affects more the performance of the mining algorithm rather than the performance of the data preparation phase; e.g., refer to tasks 1.100.all and 2.30.all. In addition, for tasks 1.100.cs88 and 2.30.cs88, the number of frequent itemsets are decided by MinSup instead of N, since MinSup is too high to include N winners in this case.

The following are two examples of interesting rules found.

1) In task 1.100.all, we found a frequent itemset $\{/SOSP99/, /SOSP99/program.html\}$ supported by 1555 accesses. This itemset can generate only one rule $\{/SOSP99/program.html \Rightarrow /SOSP99/\}$ with confidence 76.34%. From this, we can conclude that people who visit the program page for the SOSP99 conference tend to also visit the homepage of the conference; however, this is not true in the reverse order.

2) In task 1.100.cs5, we found a frequent itemset $\{/~cs5/, /~cs5/syllabus.html\}$ supported by 4161 visits. This itemset can generate two rules: $\{/~cs5/syllabus.html \Rightarrow /~cs5/\}$, with confidence 90.46%, and $\{/~cs5/ \Rightarrow /~cs5/syllabus.html\}$, with confidence 50.66. We can see that the homepage and the syllabus page of the cs5 class are often visited together. The professor can use this kind of patterns to find out how students use the course website materials.

Further research on performance using the real logs is on-going. Some interesting topics here include further tests on real logs from commercial web sites, studying differences between rules generated by using different transaction size settings, and customizing the result rule sets to focus on truly interesting rules.

Tests on Synthetic Data Sets

To assess the performance of our mining algorithm, we use synthetic datasets publicly available from IBM Quest Project Website (Agrawal et al. 96), since our web logs are not large enough to generate a truly large set of transactions. We use the notation Tx.Iy to denote a dataset in which x is the average transaction size and y is the average size of a maximal potentially frequent itemset (see (Agrawal et al. 96) for more details on the dataset generation). To keep the comparison fair, we implemented all the algorithms using the same basic data structure.

Figure 2 shows the performance comparison between MFA, (Shen et al. 98) and Apriori. We use a test data set T20.I6 that includes 100K transactions to do the test. Time(s)-axis refers to the running time in seconds, N-axis to the result size constraint, and MinSup-axis to the MinSup constraint. Tests performed using Apriori and (Shen et al. 98) are shown with some values of "N/A", since Apriori does not need a result size constraint N and (Shen et al. 98) does not use the MinSup constraint. All the other tests are performed by using MFA and each has specific N and MinSup constraints. The left part of the figure shows that when MinSup constraint becomes low, Apriori tends to run into its exponential bottleneck, and both MFA and (Shen et al. 98) do not have this performance bottleneck. The right part shows the comparison of MFA and (Shen et al. 98) in a larger scale by removing Apriori test results and from there we can see that MFA performs better than (Shen et al. 98).

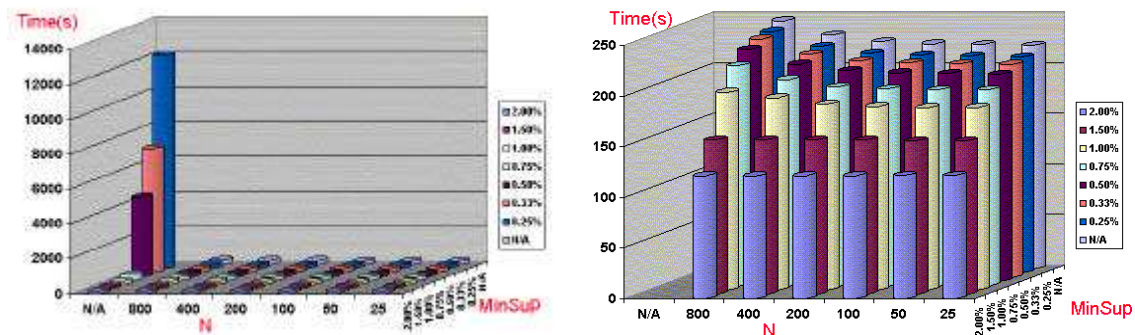


Figure 2: Performance comparison among MFA, (Shen et al. 98), and Apriori.

Figure 3 shows the scale-up performance of MFA. The left part shows the results by scaling up the result size constraint N, where we set MinSup=0.25% and use two data sets, T10.I4 and T20.I6, both containing 100K transactions. The right part shows the results by scaling up the number of items ('I' denotes the set of all involved items), where we also set N=200 and MinSup=0.25%, and use the same data sets as above. The middle part shows

the results by scaling up in the size of database D, where we set $N=200$ and $\text{MinSup}=0.25\%$ and use two groups, T10.I4 and T20.I6, of datasets with sizes (numbers of transactions) from 100K to 1M. These results show that our algorithm has good scale-up performance in terms of these 3 parameters.

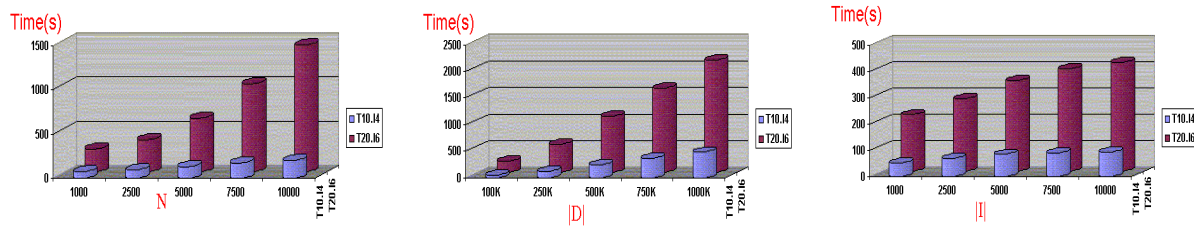


Figure 3: Scaleup Performances in terms of N (result size), |D| (database size), and |I| (number of all items).

Conclusion

This paper introduced a framework for extracting web access association patterns that are defined to be interesting by the user. We have implemented the framework in our web access pattern mining system and have evaluated this approach using both synthetic data and real web logs. Our results have shown the efficacy, efficiency and good scalability of this new mining approach. The research and development of our web log mining system are on-going. Some interesting future topics include 1) creating a data warehouse like (Joshi et al. 99) to better organize the cleaned access records and extracted mining results, and to enhance users' flexibility for accomplishing more meaningful mining tasks; 2) building a better interactive user interface and developing effective visualization tools for presenting mining results in a vivid way; 3) testing on real commercial web log data and finding new problems; and 4) developing and incorporating additional data mining and web mining techniques.

References

- Agrawal, R., Imielinski T., & Swami A. (1993). Mining Associations between Sets of Items in Massive Databases. *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Washington D.C., May 1993. 207-216.
- Agrawal R., Mannila H., Srikant R., Toivonen H., & Verkamo A.I. (1996). Fast Discovery of Association Rules. *Advances in Knowledge Discovery and Data Mining*, Chapter 12, AAAI/MIT Press, 1996.
- Garofalakis M.N., Rastogi R., Seshadri S., & Shim K. (1999). Data mining and the Web: past, present and future. *Proceedings of the 2nd international workshop on Web information and data management*, 1999. 43-47.
- Joshi K.P., Joshi A., Yesha Y., & Krishnapuram R. (1999). Warehousing and Mining Web Logs. *Proceedings of the second international workshop on on Web information and data management*, 1999. 63-68.
- Mobasher B., Jain N., Han E.H., & Srivastava J. (1996). Web Mining: Pattern Discovery from World Wide Web Transactions. Technical Report 96-050, Department of Computer Science, University of Minnesota, 1996.
- Kwedlo W., & Kretowski M. (1998). Discovery of Decision Rules from Databases: An Evolutionary Approach, *Lecture Notes in Artificial Intelligence 1510*, 1998. 370-378.
- Shafer J.C., Agrawal R., & Mehta M. (1996). SPRINT: A Scalable Parallel Classifier for Data Mining. *Proc. of the 22nd Int'l Conference on Very Large Databases*, Bombay, India, Sept. 1996.
- Shen L., Shen H., Prithard P., & Topor R. (1998). Finding the N Largest Itemsets. *Data Mining* (Editor: EBECKEN N.F.F., serve as Proc. of ICDM'98), Great Britain, WIT Pr, 1998.