Web-log Cleaning for Constructing Sequential Classifiers

Qiang Yang
Computer Science Department, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong, P.R. China qyang@cs.ust.hk

Tianyi Ian Li
School of Computing Science, Simon Fraser University, Burnaby, BC Canada V5A 1S6
tlie@cs.sfu.ca

Ke Wang
School of Computing Science, Simon Fraser University, Burnaby, BC Canada V5A 1S6
wangk@cs.sfu.ca

Abstract

With millions of web users visiting web servers each day, the web log contains valuable information about users' browsing behavior. In this work, we construct sequential classifiers for predicting the users' next visits based on the current actions using association rule mining. The domain feature of web log mining entails that we adopt a special kind of association rules we call latest-substring rules, which takes into account the temporal information as well as the correlation information. Furthermore, when constructing the classification model, we adopt a pessimistic selection method for choosing among alternative predictions. To make such prediction models useful especially for small devices with limited memory and bandwidth, we also introduce a model compression method, which removes redundant association rules from the model. We empirically show that the resulting prediction model performs very well.

1 Introduction

With the rapid growth of the Web, the web log data have become an important data source for machine learning and data mining. The web log data provide valuable information about user models. These user models are especially useful for handheld, small devices, which have only limited capacity in memory. This reality calls for two tasks. The first is to build prediction models on users' behavior that can predict user's next actions ahead of time and based on the users' current actions. These prediction models would allow better usage of network resources and the design of better user interfaces, for example, through prefetching. The second task is for researchers to build only compact models, since large models would consume too many valuable resources on such devices.

In this paper, we build association rule based models for predicting users' next visits on a web server. In the past, sequential association rules (Agrawal et. Al 1996; Agrawal et. Al 1994) have been used to capture the co-occurrence of buying different items in a supermarket shopping. Episodes were designed to capture significant patterns from sequences of events (Mannila et. Al 1995). These data mining models were not designed for the prediction task, because they do not specify how to select among multiple predictions for a given observed. The works by (Liu et. Al 1998; Wang et. Al 2000) considered using association rules for prediction by selecting rules based on confidence measures, but they did not consider the classifiers for sequential data. General classification algorithms (Quinlan 1993) were designed to deal with transaction-like data, which has a different format from the sequential data, where the concept of an attribute has to be carefully considered. In the network systems area, researchers have been using Markov models and N-grams to construct sequential classifiers. For example, (Su et. Al 2000) performed an empirical study on the tradeoffs between precision and applicability of different N-gram models, showing that longer N-gram models can make more accurate prediction than shorter ones at the expense of lower coverage. (Pitkow and Pirolli, 1999) suggested a way to make predictions based on K^{th}-order Markov models. However, these researchers did not systematically study these models from a classification point of view, and did not discuss how to compress these models into small sizes.

Thus, in this paper, we propose a new representation of association rules for sequential data called latest-substring rules. Latest-substring rules take into consideration the important domain feature in web log data that the more recent a request is, the more important is the request in predicting future requests. Substrings enforce both the adjacency information and the order information inherent in a user session in the web log, allowing for efficient mining of such rules to be done.

In building the classification model, it is also important to select among alternative predictions made by the model of a given observed sequence of actions. To this end, we introduce the pessimistic selection criterion, which prefers a prediction that maximizes the pessimistic confidence of a rule. We study the performance of this prediction models on real life web-log data. The experiments show that the latest-substring method coupled with the pessimistic selection gives better result than the traditional association rules.

Finally, it is important to note that the prediction models are most useful for empowering small devices, which have a shortage of memory and bandwidth. It is therefore important to remove redundant rules from the prediction models as much as possible. To achieve this goal, we introduce the LSIT algorithm that allows the model to be organized in a tree form. Experiments show that such models not only have smaller size but also allow efficient computation.

2 Data Cleaning for Web Log Data

In this work, we consider three realistic data sets. CSSFU (School of Computing Science in Simon Fraser University) data contains 3 days' HTTP requests to the Apache Web server serving www.cs.sfu.ca domain. The log was collected from May 1st, 2001 to May 3rd, 2001. In this period there were totally 45,637 requests. There are a total of 4,682 unique visiting IP addresses, and 5,650 sessions, and 14,664 unique pages are requested. EPA (United States Environmental Protection Agency) data contains a day's worth of all HTTP requests to the EPA WWW server located at Research Triangle Park, NC. The log was collected from 23:53:25 on Tuesday, August

29 1995 to 23:53:07 on Wednesday, August 30 1995, a total of 24 hours. In this period there were totally 47,748 requests. There are a total of 2,249 unique visiting IP addresses, 2,520 sessions, and 3,730 unique pages are requested. The NASA data is from NASA Kennedy Space Center WWW server in Florida. This data set contains one month worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida. The log was collected from 00:00:00 August 1, 1995 through 23:59:59 August 31, 1995. An example data log is shown in Figure 1.

```
kgtyk4.kj.yamagata-u.ac.jp - - [01/Aug/1995:00:00:17 -0400] "GET /
HTTP/1.0" 200    7280
kgtyk4.kj.yamagata-u.ac.jp    -    -    [01/Aug/1995:00:00:18    -0400]    "GET
/images/ksclogo -medium.gif HTTP/1.0" 200 5866
d0ucr6.fnal.gov       -       -       [01/Aug/1995:00:00:19       -0400]       "GET
/history/apollo/apollo-16/ apollo-16.html HTTP/1.0" 200
```

Figure 1. Example Web Log

Given a web log, the first step is to clean the raw data. We filter out documents that are not requested directly by users. These are image requests in the log that are retrieved automatically after accessing requests to a document containing links to these files. Their existence will not help us to do the comparison among all the different methods. We consider web log data as a sequence of distinct web pages, where subsequences, such as user sessions can be observed by unusually long gaps between consecutive requests. For example, assume that the web log consists of the following user visit sequence: (A (by user 1), B (by user 2), C (by user 2), D (by user 3), E (by user 1)) (we use "(…)" to denote a sequence of web accesses in this paper). This sequence can be divided into user sessions according to IP address: Session 1 (by user 1): (A, E); Session 2 (by user 2): (B, C); Session 3 (by user 3): (D), where each user session corresponds to a user IP address. In deciding on the boundary of the sessions, we studied the time interval distribution of successive accesses by all users, and used a constant large gap in time interval as indicators of a new session. For example, for NASA data, the gap is 2 hours.

To capture the sequential and time-limited nature of prediction, we define two windows. The first one is called *antecedent window*, which holds all visited pages within a given number of user requests and up to a current instant in time. A second window, called the *consequent window*, holds all future visited pages within a number of user requests from the current time instant. In subsequent discussions, we will refer to the antecedent window as *W1*, and the consequent window as *W2*. Intuitively, a certain pattern of web pages already occurring in an antecedent window could be used to determine which documents are going to occur in the consequent window.

The moving windows define a table in which data mining can occur. Each row of the table corresponds to the URL's captured by each pair of moving windows. Figure 2 shows an example of such a table corresponding to the sequence (A, B, C, A, C, D, G), where the size of W1 is three and the size of W2 is two. In this table, under W1, A1, A2 and A3 denote the locations of the last three objects requested in the antecedent window, and P1 and P2 are the two objects in the consequent window.

W1			W2	
A1	**A2**	**A3**	**P1**	**P2**
A	B	C	A	C
B	C	A	C	D
C	A	C	D	G

Figure 2. A portion of the Log Table extracted by a moving window pair of size [2, 2]

3 Building Sequential Association-Rule Based Classifiers

Having the relational representation of the logs as in Figure 2, we can construct rules of the form LHS$\rightarrow$RHS. To focus our attention, we restrict the RHS in the following way. Let {U1, U2, …Un} be the candidate URL's for the RHS that can be predicted based on the same LHS. We build a rule LHS$\rightarrow$ Uk where the pair {LHS, Uk} occurs most frequently in the rows of the table among all Ui's in the set {U1, U2, …Un}. Ties are broken arbitrarily. This is the rule with the highest support among all LHS$\rightarrow$Ui rules.

We restrict the rules to be the *latest-substring rules*. These rules not only take into account the order and adjacency information, but also the *recency* information about the LHS string. In this representation, only the substrings ending in the current time (which corresponds to the end of the window W1) qualify to be the LHS of a rule. A substring of a string is a sequence of symbols that occur in both the same relative order and adjacency. The latest-substring rules are also known as hybrid n-gram rules in some literature. For example, Figure 3 shows the latest-substring rules example.

W1	W2	Latest substring Rules
A, B, C	D	<A, B, C> $\rightarrow$ D, <B, C> $\rightarrow$ D, <C> $\rightarrow$ D

Figure 3: Latest-substring rules

Related to previous works, the latest-substring rules could also be considered as the union of Nth-order Markov models (Nicholson et. Al 1998), where N covers different orders up to the length of *W1*. Therefore, it is more general than the N-gram models or Nth-order Markov models. However, through our other experiments, we have found out that the Markov models' performance drops when N exceeds a certain threshold, but the latest-substring method that considers multiple Nth-order models experience a monotonically increasing precision curve.

When building the classification model, we add a default rule that captures all cases where no rule in the rule set applies; when no LHS of all rules apply to a given observed sequence of URL's, the default rule always applies. For example, a default rule can simply be the most frequently requested page in the training web log.

For each rule of the form LHS$\rightarrow$ RHS, we define the *support* and *confidence* as follows

$$\sup = \frac{count(LHS, RHS)}{count(Table)} \tag{1}$$

$$conf = \frac{\sup(LHS, RHS)}{\sup(LHS)} \tag{2}$$

In the equations above, the function *count*(*Table*) returns the number of rows in the log table, and sup(LHS, RHS) counts the number of times an itemset occurs in the data.
In classification, our goal is to output the best guess on a class based on a given observation. When applying the latest-substring rule to an observed case, we can possibly derive multiple predictions from different rules that match. Therefore, we need a way to select among all rules that apply.

A traditional and elegant way to improve classifiers is to use Pessimistic-Error Estimate (Quinlan 1993). Instead of using the measurement of confidence, we define a new measurement called pessimistic confidence. We denote the number of incorrectly classified cases as *E* and the total number of cases classified by the rule as *N*. And, we define *pessimistic confidence* as

$$conf_p = 1 - \frac{U_{CF}(E,N)}{N} \tag{3}$$

For a given confidence level 1-CF, the upper limit on the error rate over the whole population is $U_{CF}(E,N)$ with confidence value equal to CF/2. In pessimistic selection, we only use the upper limit of the error rate as the estimate on potential error rate in test data, because this method is always *pessimistic* about the accuracy of classification model. Therefore, it always expects a higher error rate using the classifier on unknown testing data.

For example, consider the following rules.

Rule 1: (A) → B with confidence 100%, N=1, E=0,
Rule 2: (D) → E with confidence 80%, N=100, E=20;

Suppose that Rule2 has been applied to predict on 100 cases in the training data set, of which 80 are incorrectly predicted. For a confidence level 75%, the estimated upper limit (or pessimistic limit) of the real error rate is $U_{0.25}$ (100,20). Computing the pessimistic confidence on both rules, we get:

For Rule 1: pessimistic confidence = 1 - $U_{0.25}$ (1, 0) = 25%,
For Rule 2: pessimistic confidence = 1 - $U_{0.25}$ (100, 20) = 76.57%

The pessimistic-selection method picks a rule with the *highest* pessimistic confidence in all the applicable rules. Ties are broken arbitrarily. In this case, Rule 2 is regarded as more reliable. Thus, Rule 2 is selected with E as the prediction.

4 Latest-Substring Index Tree (LSIT)

Although we have now a way to build the prediction model using the latest-substring method, it is not difficult to see that with a large antecedent window, the resultant rule set will grow quickly.

The dramatic increase in the number of association rules can significantly offset the benefit of getting a higher prediction precision from longer association rules, and limits the applicability of our prediction models to applications where memory and bandwidth are small. This is especially true for hand-held devices. To address these problems, we make use of the desirable properties of latest-substring rules. We propose a tree-like structure called *Latest-Substring Index Tree (LSIT)* to make the web-document prediction model more efficient to apply and less memory consuming.

To illustrate, consider the following example. Suppose that we have two rules being applicable to the testing case <B, C> $\rightarrow$?

Rule 1:	<A, B, C> $\rightarrow$ D	(pessimistic confidence = 50%)
Rule 2:	<B, C> $\rightarrow$ E	(pessimistic confidence = 70%)

Because Rule 2's pessimistic confidence is higher than Rule 1, we will select Rule 2 as the best rule, resulting in the prediction E. A careful examination indicates that in all cases where Rule 1 can be applied, Rule 2 can also be applied. In this case, it is obvious that Rule 1 will never be used because Rule 2 is more general and has a higher confidence value than Rule 1. Thus, there is no need to keep Rule 1 in the prediction rule set. If we examine relations among the rules in the model, and we remove some rules similar to Rule1, we will have a smaller rule set without loss of quality.

We organize the association rules in the prediction model into a tree structure we call Latest-Substring Index Tree (LSIT). For each latest-substring rule R1 whose LHS is <A_1, A_2, …, A_i>, another rule R2 with a LHS = <A_2, …, A_i> is called R1's *direct parent rule*; R2 is the direct child rule of R1. An important property of the latest-substring rule is that a non-default rule has a unique direct parent rule. In the extreme case, the direct parent rule of all rules with a singleton LHS is the default rule '$\varnothing \rightarrow$ P', where P is a most popular page in the web log. A rule can have zero or more direct children rules. This parent-children relationship is a tree-like structure.

We construct the LSIT according to the following relations:

- Each rule is represented by a node in the LSIT;
- The node representing the direct parent rule is the parent node of the node(s) representing the direct children rule(s);
- The root of the LSIT representing the default rule.

We give an example in Figure 4 to illustrate the five-node LSIT mapped from a five-rule prediction model.

Rules	Pessimistic confidence
$\varnothing \rightarrow$ M	50%
<C> $\rightarrow$ N	40%
<A> $\rightarrow$ M	30%
<B, C> $\rightarrow$ P	70%
<C, C> $\rightarrow$ Q	80%
<E, A> $\rightarrow$ R	40%

Figure 4: Example rule set

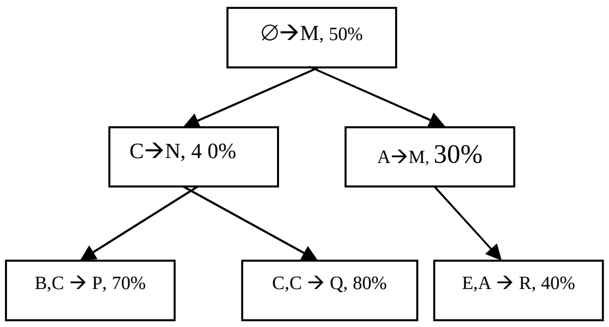

Figure 5: LSIT Tree from Figure 4

An important property of the LSIT tree is that some nodes can be pruned without affecting the performance of the model. The pruning process traverses the tree using a post-order traversal. If a node has a lower pessimistic confidence than that of its direct parent nodes, or it predicts the same class as its direct parent node, it can be pruned and all of its children nodes (if any) promoted to be the children of its direct parent node. After one node has been eliminated, the algorithm will traverse the new sub-tree rooted on its parent node, to examine whether there are more nodes to be pruned.

For example, in Figure 5, the node <C> → N can be pruned since its pessimistic confidence is lower than its direct parent. The node <A> → M can be pruned because its parent node predicts the same class as it does. After these two nodes are pruned, the node <E, A> → R can be pruned because of its lower pessimistic error. After pruning, the structure of the LSIT is simpler, as we can see in Figure 6.

After the pruning is completed for the whole LSIT tree, the pessimistic confidence value for each node becomes useless, because the children will always have a higher pessimistic confidence value than their parents. Then, we mark the pointers going from parents to their children by the extra URL's in the children. These extra URL's are used to label the edges, and internal nodes of the LSIT tree is no longer marked by the LHS. Figure 5.3 shows the final structure of the LSIT.

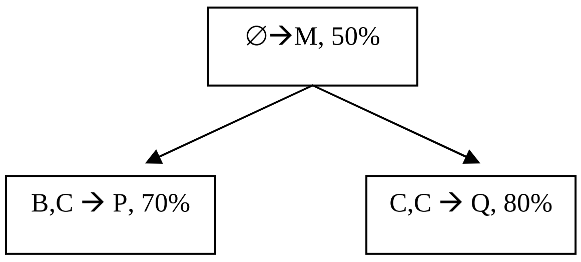

Figure 6. LSIT after pruning

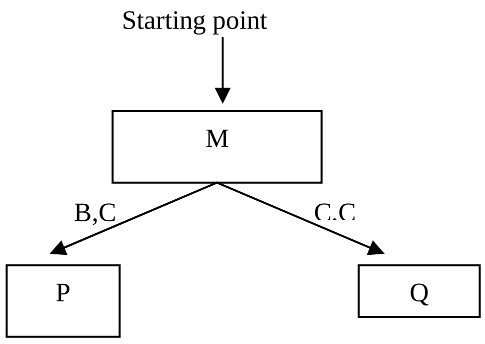

Figure 7: LSIT after indexing

Now we can use the resultant LSIT for prediction. If a new test case comes to the LSIT, it is not necessary to collect all the applicable rules. Instead, we use the current accessed page as the index, then the last visited page, etc, until the LSIT cannot find the next index in the tree. The last node it reaches in its path should be used to make the prediction. For example, referring to the LSIT in Figure 7, if the testing case is <C> → ?, the classifier will index on C, but unable to find the correct path indexed as 'C'. In this case, the rule from the LSIT is located at the root 'M'. Thus the predicted class is M.

Continuing with the example, if the testing case is <A, B, C> → ?, the classifier will first index on 'C', if a path could not be found, it will next index on 'B, C', to find the node 'P'. Since 'P' is already a leaf node, there is no reason to extend the indexing process. Therefore, the prediction will be P.

5. Evaluation

The structure of LSIT reduces the size and improves the accuracy of the classifier without sacrificing the potential accuracy of the prediction models. To evaluate the size reduction, we compared the sizes of latest-substring rule sets before and after the LSIT pruning; the rules in the rule sets are extracted from antecedent windows of different sizes. Experiments are based on NASA data. Results are shown in Figure 9.

From Figure8, we clearly see that the LSIT pruning reduces the memory usage of the prediction models. Before pruning, with the W1 size increases, a large number of specific rules with low pessimistic confidence are accepted into the classifier. However, only a few of them are accepted in terms of pessimistic confidence. Pruning of these rules by comparing the pessimistic confidence gives us a much more compact model. Only the useful rules are kept in the model, and about 4/5 of the rules have been eliminated.

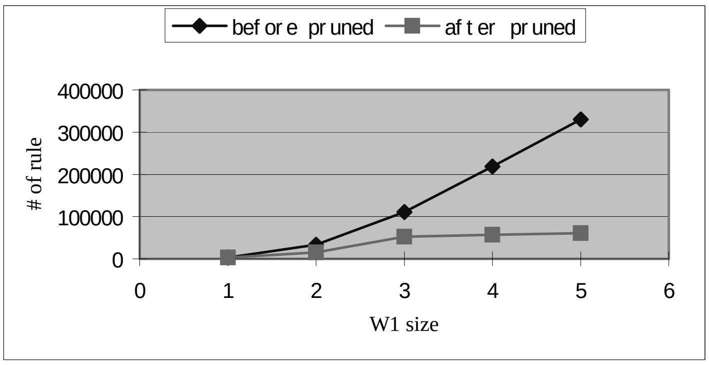

Figure 8 Reduced rule set size after LSIT pruning

5 Conclusions and Future Work

In this paper, we presented sequential classifiers for predicting the users' next visits based on the current actions using association rule mining. The domain feature of web log mining entails that we adopt a special kind of association rules we call latest-substring rules, which takes into account the temporal information as well as the correlation information. Furthermore, when constructing the classification model, we adopt a pessimistic selection method for choosing among alternative predictions. To make such prediction models useful especially for small devices with limited memory and bandwidth, we also introduce a model compression method, which removes redundant association rules from the model. Our empirical results show that the resulting prediction model performs well under testing in realistic data.

References

M.Arlitt, R.Friedrich, L.Cherkasova, J.Dilley, and T.Jin. 1999. Evaluating Content Management Techniques for Web Proxy Caches. *HP Technical Report*, Palo Alto, Apr. 1999.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo. 1996. Fast discovery of association rules. *Advances in knowledge discovery and data mining*. Pp. 307-328, AAAI/MIT Press, 1996.

R. Agrawal and R. Srikant. 1994. Fast algorithm for mining association rules. *Proceedings of the Twentieth International Conference on Very Large Databases.* 1994. pp 487-499

R. Agrawal and R. Srikant. 1995. *Mining sequential patterns*. In Proceedings of the 1995 Int. Conf. Data Engineering, Taipei, Taiwan, March 1995. Pages 3 -14.

J. Borges and M. Levene, 2000. A Heuristic to Capture Longer User Web Navigation Patterns. In *Proc. the First Int'l Conf. on Electronic Commerce and Web Technologies*, Greenwich, U.K., September 2000

M. Chen, J. S. Park, and P. S. Yu, 1996. Data Mining for Path Traversal Patterns in a Web Environment. In *Proc. the 16th Conference on Distributed Computing Systems*, May 1996, pp. 385-392.

B.Liu, W.Hsu, and Y.Ma. 1998. Integrating Classification and Association Mining. In *Proc. of the Fourth Int'l Conf. on Knowledge Discovery and Data Mining (KDD-98)*, New York, 1998, pp. 80-86.

A. E. Nicholson, I. Zukerman, and D. W. Albrecht. 1998. A Decision-theoretic Approach for Pre-sending Information on the WWW. In *PRICAI'98 - Proc. the Fifth Pacific Rim Int'l Conf. on Artificial Intelligence,* Singapore, page 575-586.

M. Perkowitz and O. Etzioni. 2000. Towards Adaptive Web Sites: Conceptual Framework and Case Study. *Artificial Intelligence Journal*, 118(2000). PP. 245-275.

J.Pitkow and P.Pirolli, 1999. Mining Longest Repeating Subsequences to Predict World Wide Web Surfing. In *Second USENIX Symposium on Internet Technologies and Systems*, Boulder, C0, 1999.

J.R. Quinlan, 1993. C4.5: Programs for Machine Learning. *Morgan Kaufmann*, 1993.

J. Srivastava, R. Cooley, M. Deshpande, and P. Tan 2000. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations*, 1(2), 2000, pp. 12-23.

S. Schechter, M. Krishnan, and M. D. Smith, 1998. Using Path Profiles to Predict HTTP Requests. In Proc. *7th International World Wide Web Conference*, Brisbane, Qld., Australia, April 1998, pp. 457--467.

Z. Su, Q. Yang, Y. Lu, and H. Zhang. 2000 Whatnext: A Prediction System for Web Requests Using N-gram Sequence Models. In *Proc. of the First Int'l Conf. on Web Information Systems and Engineering Conference*, Hong Kong June 2000, pp. 200-207.

Z. Su, Q. Yang, and H. Zhang 2000 A Prediction System for Multimedia Pre-fetching in Internet. In *Proc. 2000 Int'l ACM Conf. on Multimedia*, Los Angeles, California, *2000*.

K. Wang, Y. He, and J. Han 2000. Mining Frequent Itemsets Using Support Constraints. In *Proc. 2000 Int. Conf. on Very Large Data Bases (VLDB'00)*, Cairo, Egypt, September 2000, page 43-52.