Mining Association Patterns in Web Usage Data

Pang-Ning Tan†, Vipin Kumar†

Abstract—**This paper presents our work in using data mining techniques to discover interesting association patterns from Web usage data. In particular, we address some of the key issues involved in preprocessing and mining the Web association patterns. For preprocessing, we describe how to build accurate classification models to eliminate superfluous sessions created by Web robots. For mining, we develop a new technique called indirect association to capture interesting negative association between different groups of Web users.**

Keywords— **Web Usage Mining, Association, Classification.**

I. Introduction

The World-Wide Web has left a profound effect on all aspects of the society. It is an important medium for disseminating information as well as a valuable resource for gathering information. The click-streams generated by Web users while surfing the Web provide a great opportunity for e-commerce organizations to collect information about their online users. As the number of online users grow, so does the volume of click-stream data collected at the Web servers. This has fueled a tremendous amount of interest in applying data mining techniques to discover hidden patterns from the Web data [1]. One interesting type of pattern is the *Web association pattern*, which contains information about what are the pages frequently accessed together in the same session. An example of such pattern could be that 60% of visitors who access the Sports Web page on an online newspaper Web site also looked at the TV guide. Knowledge derived from the Web association patterns can be used for the following purposes:

- to assist Web site administrators in re-structuring the organization of their Web site by adding, deleting, promoting or demoting hyperlinks.
- to predict what is the next Web page requested by an ongoing session. Such information can be used to develop caching policies at proxy servers and to pre-generate HTML pages so that the latency caused by dynamic Web page generation can be reduced.
- to collect business intelligence about the behavior of Web site visitors. Such information can be used to develop effective marketing promotions targeted towards certain group of Web users.

In this paper, we will highlight some of the key issues involved in extracting association patterns from click-stream data. Mining Web association patterns is more than just transforming the raw click-streams into a suitable data format to enable the application of existing association mining algorithms. First, it requires considerable preprocessing effort to create reliable Web sessions and to eliminate superfluous sessions (such as those by Web robots). Second, the current association mining framework may not adequately capture all the interesting relationships in Web data. This is because the existing framework has focused primarily on finding patterns that occur frequently in the data, i.e. those with sufficiently high *support*[1]. Any patterns that do not have enough support are assumed to be statistically insignificant and therefore, eliminated. However, this may end up removing useful information such as patterns involving negatively-correlated pages. The negatively-correlated patterns can be quite useful even though the pages are seldom visited together because they may represent the distinct browsing behavior for different groups of Web users. Third, association pattern was originally proposed to analyze customer transactions in the market-basket domain, where there is very little constraints governing what type of items should or should not be bought together in the same transaction. In the Web domain, there is evidence that the traversal path of Web users can be influenced by the constraints imposed by the Web site structure. For instance, Web pages that are located closer to the home page tend to have higher hit rates compared to those located further away (Figure 1). In addition, Web pages that have many outgoing links (i.e., hub pages) also tend to have higher support, as shown in Figure 2. As a consequence, the majority of the Web association patterns would contain these so-called navigation pages (the home page and hub pages). A navigation page is an index page that has many outgoing links to guide the users to pages that contain the actual information content the Web site is providing. It is not quite clear how to effectively handle patterns containing the navigation pages. In most cases, these pages are not as interesting as the actual content pages. However, eliminating them from the data is too drastic because some of them could be informative when used together with the content pages. Keeping all of them could be problematic too because we could end up with too many patterns involving the frequent co-occurrences of navigation pages.

This paper describes several techniques we have developed to address some of the issues mentioned above. First, we have built highly-accurate classification models for detecting Web robot sessions. Second, we have introduced a new type of pattern called indirect association to capture interesting negative associations in the Web data. The impact of site structure on the discovered patterns can also be addressed using the indirect association framework.

The remainder of this paper is organized in the following way. In Section II, we present an overview of the association mining problem. Next, we will discuss some of

†Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455

[1]The support of a pattern is the fraction of data set for which the pattern is observed.

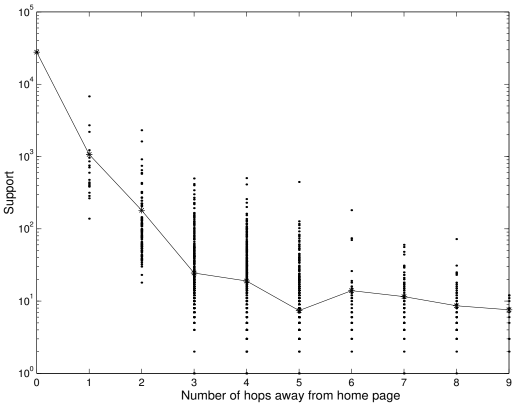

Fig. 1. Average support of a Web page against distance (measured in the terms of number of hops away) from the home page. The result is obtained from the Web logs for the University of Minnesota Computer Science department Web site.

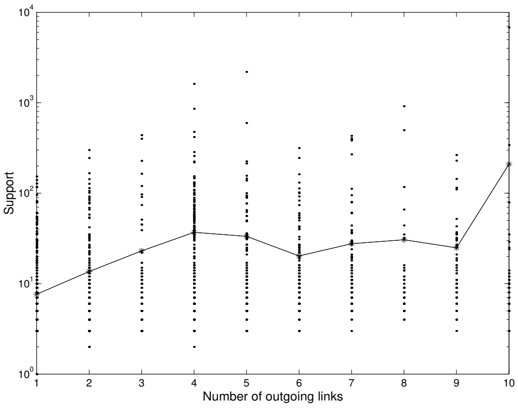

Fig. 2. Average support of a Web page against the number of outgoing links. The result is obtained from the Web logs for the University of Minnesota Computer Science department Web site.

the preprocessing issues, focused mainly on the problem of eliminating Web robot sessions. We then introduce the concept of indirect association and show how it can be used to capture interesting negative associations and address the site structure issues. Finally, we conclude with a summary of our experimental results.

II. Background

A. Web Usage Data

A Web server log is an important data source for performing Web Usage Mining because it explicitly records the browsing history of site visitors. Figure 3 shows a typical Web server log, which contains information such as the IP address and user agent of the Web client, the URI (Uniform Resource Identifier) of the requested and the referrer pages, the HTTP protocol and request method used, the timestamp and status of each request, and the size of the file that was transferred from the Web server to the client. Entries in the Web log are ordered according to their timestamps.

TABLE I
Representing Web sessions as transactions.

Session	Web Pages
1	Homepage, Metro
2	Homepage, Metro, World, Sports
3	Homepage, Sports, TV, Metro
4	Homepage, Sports, TV, Entertainment
5	Homepage, Metro, Weather

B. Overview of Association Rules

Association rule mining [2] has emerged as one of the most important techniques in data mining because an association rule can capture the significant relationships among attributes of a data set. Formally, the association rule mining problem can be stated as follows: Let I denotes the set of all items, and T is the set of database transactions, where $\forall t \in T: \; t \subseteq I$. An association rule is an implication expression of the form $A \longrightarrow B$, where $A, B \subset I$ and $A \cap B = \emptyset$. (In association rule terminology, A and B are called itemsets.) This rule simply says that if all the items in itemset A are present in a given transaction, then there is a strong possibility that the transaction also contains all the items in B. The term association pattern used in this paper refers to all forms of associations, including itemsets and association rules.

The strength of an association rule is measured in terms of two empirical metrics: *support* s and *confidence* c. The rule $A \longrightarrow B$ has support s if s% of transactions in T contains $A \cup B$. The rule holds with confidence c if c% of the transactions that contain A also contain B. The definition of these metrics are summarized below:

$$\begin{aligned} s(A \longrightarrow B) &= \frac{\sigma(A \cup B)}{|T|} \\ c(A \longrightarrow B) &= \frac{\sigma(A \cup B)}{\sigma(A)} \end{aligned} \quad (1)$$

where $\sigma(A)$ denotes the number of transactions that contain the itemset A. Given a set of transactions T, the goal of association rule mining is to generate all rules with support greater than a user-specified minimum support threshold ($minsup$) and confidence greater than a minimum confidence threshold ($minconf$). An itemset that has support greater than $minsup$ is also known as a frequent itemset.

Example 1: Consider the Web sessions shown in Table I. This data set has 5 transactions (sessions) and 7 items (pages). The support for Sports page is 3/5 because there are 3 sessions containing this page. Similarly the support for TV page is 2/5 while the joint support for the itemset {Sports, TV} is also 2/5. Thus, the association rule TV $\longrightarrow$ Sports has a support of 2/5 (40%) and confidence 2/2 (100%) according to Equation 1.

The association rule discovery problem is often decomposed into two steps: (1) to find all the itemsets having support greater than $minsup$, and (2) to extract rules with

#	IP Address	Userid	Timestamp	Method/URL/Protocol	Status	Size	Referrer	User Agent
1	160.94.178.203	-	[01/Dec/2001:00:00:01 -0600]	"GET / HTTP/1.1"	200	1234	-	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
2	160.94.178.203	-	[01/Dec/2001:00:00:27 -0600]	"GET /sports.html HTTP/1.1"	200	5600	/	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
3	128.101.38.52	-	[01/Dec/2001:00:00:32 -0600]	"GET /world.html HTTP/1.1"	200	6170	/	Mozilla/4.7 [en] (X11; I; Linux 2.4.0 i686)
4	160.94.178.146	-	[01/Dec/2001:00:01:02 -0600]	"GET / HTTP/1.0"	200	1234	-	Mozilla/4.7 [en] (X11; I; Linux 2.4.0 i686)
5	160.94.178.203	-	[01/Dec/2001:00:01:08 -0600]	"GET /sports/nba.html HTTP/1.1"	200	4223	/sports.html	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
6	128.101.38.52	-	[01/Dec/2001:00:01:15 -0600]	"GET /world/mideast.html HTTP/1.1"	200	1310	/editor.html	Mozilla/4.7 [en] (X11; I; Linux 2.4.0 i686)
7	160.94.178.146	-	[01/Dec/2001:00:01:32 -0600]	"GET /metro.html HTTP/1.0"	200	2010	/	Mozilla/4.7 [en] (X11; I; Linux 2.4.0 i686)
8	160.94.178.203	-	[01/Dec/2001:00:01:45 -0600]	"GET /sports/nba/results.html HTTP/1.1"	200	910	/sports/nba.html	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
9	160.94.178.146	-	[01/Dec/2001:00:02:22 -0600]	"GET /weather.html HTTP/1.0"	200	1005	/	Mozilla/4.7 [en] (X11; I; Linux 2.4.0 i686)
10	160.94.178.203	-	[01/Dec/2001:00:02:35 -0600]	"GET /tv.html HTTP/1.1"	200	8100	/	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)

Fig. 3. Example of a Web server log.

1. $F_1 = \{$ frequent 1-itemsets$\}$;
2. **for** ($k = 2$; $F_{k-1} \neq \phi$; k++) {
3. $\quad C_k$ = apriori_gen(F_{k-1})
4. $\quad$ **for** all transactions $t \in T$ {
5. $\qquad$ subset(C_k, t)
6. $\quad$ }
7. $\quad F_k = \{c \in C_k \mid \text{c.count} \geq \text{minsup}\}$
8. }
9. Answer = $\bigcup F_k$

TABLE II
Apriori Algorithm

confidence greater than *minconf* from the frequent itemsets generated in step 1.[2] The computational time for the first step can be quite expensive, due to the exponential size of the search space. (The total size of the search space is $2^{|I|}$.) The first algorithm to handle the search problem elegantly was the Apriori algorithm [3]. It uses the following heuristic to reduce the computational time for itemset generation: if an itemset of size k is frequent, then all of its subsets must also be frequent. This is often called the anti-monotone property of support.

A brief summary of the Apriori algorithm is presented in Table II. The Apriori algorithm makes multiple passes over the entire database. During pass k, the algorithm would generate candidate itemsets of size k by extending the frequent itemsets of size $k-1$ generated in the previous pass (Step 3). For example, if $\{A, B, C\}$ and $\{A, B, D\}$ are frequent itemsets of size-3, then the apriori-gen routine will generate a candidate itemset $\{A, B, C, D\}$. Next, it will scan the database to determine the actual support count of each candidate itemset (Steps 4 through 6). At the end of pass k, any candidate itemsets having support below *minsup* will be discarded (Step 7), and the entire process is repeated until no new frequent itemsets are generated.

III. Preprocessing Issue

A. Preliminary

A typical way to generate Web association patterns involves the following tasks: (1) preprocess the click-streams into Web sessions, (2) convert the Web sessions into record-based transaction format (Table I), and (3) run existing algorithm (such as Apriori) on the Web transactions. Identifying individual Web sessions from the Web server log can be quite a challenging task if client-side tracking techniques (such as Java applets and cookies) are not available. We have proposed a robust session identification technique for Web server logs in [4].

Once the Web sessions have been created, transforming it into record-based transaction format is a trivial task. However, before applying the standard association mining algorithm, one should be wary of superfluous sessions created by Web robots. Web robots are software programs that automatically traverse the hyperlink structure of the World Wide Web in order to locate and retrieve information. These robots are deployed to scour the Web for various purposes. For instance, Internet search engines such as Google and Altavista rely on the documents retrieved by Web robots to build their internal search index databases while Web administrators employ Web robots to perform site maintenance tasks such as checking for broken hyperlinks.

The presence of Web robots can distort the overall distribution of Web page accesses. For example, Figure 4 shows the total number of sessions and HTML pages requested at the University of Minnesota Computer Science department Web site between the period of January 1, 2001 and January 31, 2001. On average, about 5% of the total sessions are due to visits by Web robots. However, these sessions may account for as high as 85% of the total number of HTML pages requested daily. If the robot sessions are not identified and eliminated, analysts may end up making wrong inferences about the behavior of their site visitors.

To illustrate this point, consider the example of browsing patterns for a human user and a Web robot shown in Figure 5. If most of the requests for pages A and D are due to Web robots, then one would end up thinking that users who view page A are also interested in page D even though such pattern is an artifact of the Web robot accesses. Thus, it is important to eliminate the Web robot sessions from the data set prior to pattern discovery.

There are other reasons why it is important to identify visits by Web robots and distinguish them from other users. For example, e-commerce retailers are particularly concerned about the unauthorized deployment of Web robots

[2] Note that an association rule is a binary partitioning of a frequent itemset.

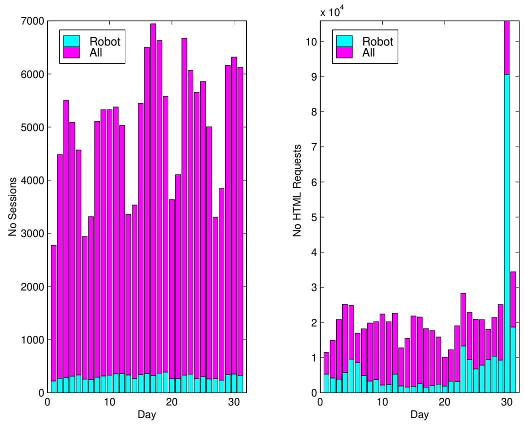

Fig. 4. Summary of daily Web traffic at the University of Minnesota Computer Science department Web server. The above figure shows the fraction of total sessions and HTML requests due to Web robots. The anomaly on day 30 for the right-hand figure is due to HTML requests from a site mapping robot called linbot.

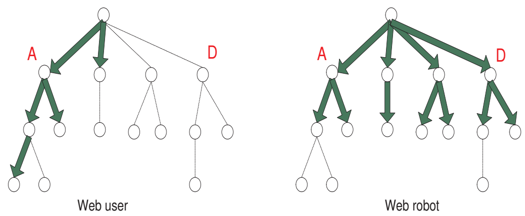

Fig. 5. Distortion of Web traffic distribution due to accesses by Web robots.

for gathering business intelligence at their Web sites. In such a situation, the e-commerce site may want to block HTTP requests coming from the unauthorized robots [5]. The deployment of Web robots also comes at the expense of other users because they often consume considerable network bandwidth and server resources. In this case, it would be desirable to detect the robots and reduce their priority of service immediately.

B. Web Robot Detection

Although Web robot detection is a widely recognized problem, there are very few papers published in this area. A standard way to detect robots is by comparing their IP address and user agent fields against those of known robots as shown in Table III [6], [7]. However, since Web robots can be easily constructed and deployed, it is impossible to keep a comprehensive list of IP addresses and user agents for all robots. This problem is further exacerbated by robots that attempt to disguise their identities by using user agents that look very similar to conventional browsers (e.g. the last entry of Table III). As a result, the standard approach will fail to detect the presence of such camouflaging robots.

Another way to detect robots is by looking at sessions that access a special file called `robots.txt`. According to the Robot Exclusion Standard [8], [9], whenever a robot visits a Web site, it should first examine the content of the

TABLE III

List of IP addresses and user agents for several known Web clients.

Client's Type	IP address	User agent
Netscape browser	160.94.178.152	Mozilla/4.7 [en] (X11;I; Linux 2.2.14-5.0 i686)
IE browser	160.94.178.205	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)
Search Engine	199.172.149.184	ArchitextSpider
Email Harvester	4.41.77.204	EmailSiphon
Link Checker	130.237.234.90	LinkChecker/1.0
Search Engine (looksmart.com)	207.138.42.10	Mozilla/4.5 [en] (Win95; I)

`robots.txt` file for that particular site. The purpose of this file is to allow Web administrators to specify which part of their Web site is off-limits to visiting robots. For example, the following entry in the `robots.txt` file forbids all robots from accessing the file `A.html`.

```
User-agent: *
Disallow: /A.html
```

This suggests that Web robots can be easily detected by examining sessions that access the `robots.txt` file. Unfortunately, this is not a full-proof technique because compliance to the Robot Exclusion standard is voluntary, and many robot designers do not follow the proposed standard.

Various guidelines have also been proposed to regulate the behavior of Web robots at a Web site. These guidelines suggest that an ethical robot should (1) moderate its rate of information acquisition, (2) operate only when the server is lightly loaded (e.g. at night), and (3) use the HEAD request method, whenever possible. The request method of an HTTP message (e.g. GET, HEAD and POST) determines what type of action the Web server should perform on the resource requested by the Web client. Typically, a Web server would respond to a GET request by sending back a message, containing some header information along with a message body, which contains the requested file. On the contrary, the response to a HEAD request contains only the message header, and thus, incurs less communication overhead. This is why Web robots are often encouraged to use the HEAD request method. Link checker robots, for example, use the HEAD request primarily to verify whether a hyperlink still exists. Another way to detect robots is to look for sessions that have an unusually large number of requests with unassigned referrer fields. The referrer field is provided by the HTTP protocol to allow a Web client (particularly, a browser) to specify the address of the Web page that contains the link the client followed in order to reach the current requested page. Since most robots do not assign any value to their referrer fields, these values appear as "-" in the Web server logs. Once again, both heuristics are not entirely reliable because non-robots can also generate HEAD request messages (e.g. when proxy servers attempt to validate their cache contents) and HTTP messages with unassigned referrer values (e.g. when a user clicks on a bookmarked page or types in a new URI in the address window). This suggests that a more robust technique is needed to identify visits by camouflaging or previously unknown Web robots.

C. Building Classification Models for Robot Detection

In this paper, we offer an alternative solution to the robot detection problem by building classification models to distinguish robot from non-robot sessions. Our main assumption is that the navigational patterns for Web robots are inherently different than those of Web users.

Building classification models from Web usage data is a challenging problem due to the following reasons. First, the click-stream data is sequential and temporal in nature. Therefore, the models must take into account both the static and dynamic properties of each session. Furthermore, since the length of a Web session varies from one session to another, it is not sufficient to generate a single classification model for all the sessions. Instead, the models must be built incrementally, after each request made by the Web client. Second, it is not trivial to construct a reliable training set for the classification task. One must identify the useful features for discriminating robot from non-robot sessions and determine the appropriate class label for each session. Manual assignment of the class labels can be a laborious task, while an automated procedure may introduce labeling errors into the training data. Third, it would be desirable to detect the Web robots swiftly, after looking at just a few requests. This is a formidable goal because the smaller the number of requests of a session, the less information it contains to make accurate prediction. Hence, building classification models for Web robots involve a tradeoff between early detection and model accuracy.

The features used for building our classification models are shown in Table IV. We can divide these features into four categories: (1) temporal features, (2) page features, (3) communication features, and (4) path features. The temporal features such as $Night$, $totalTime$, $avgTime$ and $stdevTime$ describe the properties of a session in terms of their request times. The page features such as totalPages, % Image and % HTML characterize the properties of the pages accessed by a Web session, while the communication features such as *Repeated*, *Error*, GET and HEAD describe the properties of the HTTP messages exchanged between the Web server and the client. For example, $MultiIP$ and $MultiAgent$ are two communication features that indicate whether a session uses multiple IP addresses or user agents.

The path features, such as *width* and *depth*, describe the characteristics of the navigational paths of a Web user or robot. To derive these features, we need to construct a directional graph representing the HTML pages requested by the session. The *width* feature measures the number of leaf nodes generated in the graph while *depth* measures the maximum depth of the tree(s) within the graph. For example, if a session consists of the following set of (request, referrer) pages, { (/A,-), (/A/B,/A), (/A/B/C,/A/B)}, then the *width* of the session is 1 while its *depth* is 3. As another example, a session that contains the following set of (request, referrer) pages, {(/A,-) (/A/B,/A), (/C,-) (/D,-) }, will have a *width* of 3 and a *depth* of 2.

Not all features given in Table IV are used for building the classification models. Three of them (robots.txt, referrer= "−" and HEAD) are used to assign the class label of a session, while a fourth feature (*length*) is used to create data sets of various session lengths. The following scheme is used to determine the class label of a session:

1. If a session s contains a request for the `robots.txt` file, then it is declared to be a robot session (denoted by $class = 1$).
2. We label the rest of the sessions according to their user agents. If the user agent of the session matches that of a known robot, we label the session as a Web robot. To do this, we need to have a comprehensive list of user agents for all the robots and non-robots found in our data set.
3. We use the referrer = "−" and HEAD features to identify new robot sessions in the training and test data sets. If all the HTTP request messages of a session use the HEAD request method or if all of their referrer values are empty, then the session will most likely belong to a robot, as long as the number of such requests is sufficiently high.[3] We use these heuristics to detect new robots and update the list of user agents described above.

After deriving the session features, classification models are built using the well-known C4.5 decision tree algorithm [10]. There are various metrics we can use to evaluate the performance of our classification models. Accuracy is a reasonable metric, as long as the data set is evenly distributed between robots and non-robots. Otherwise, we need to compensate for the imbalanced class distribution via stratification, or by using techniques such as bagging and boosting. Two other metrics used for evaluating the performance of our classification models are:

$$\text{recall}, r = \frac{\text{no of robot sessions found correctly}}{\text{total no of actual robot sessions}} \quad (2)$$

$$\text{precision}, p = \frac{\text{no of robot sessions found correctly}}{\text{total no of predicted robot sessions}} . \quad (3)$$

Since we are interested in detecting Web robot activities as early as possible, it is necessary to build the classification models incrementally (after each request) and determine the minimum number of requests needed to identify robot sessions accurately. In order to do this, we have to create data sets of feature vectors at various session lengths (i.e., number of HTML requests). For example, the data set for one request is created from all the sessions because the minimum length of a session is one. If a session contains only one request, we use all the information within the session to create its feature vector. However, if the session length is greater than one, we derive the values of its feature vector using only the session information up to its first HTML request and ignore the rest of the pages accessed after the first request. For the data set with session length equals to two, we discard sessions that have only one HTML request, and consider only sessions that have

[3] A Web browser can generate requests with empty referrer fields when a user submits a new URI from the address window or clicks on a bookmarked page (this type of request is known as a user-input click). However, if the number of requests with empty referrer fields is high, they are more likely created by robots than browsers.

TABLE IV

Summary of features derived from the Web sessions. These features are either used to determine the class labels (denoted as Label) or to build the classification models (denoted as Model).

Id	Feature	Remark	Purpose
1	totalPages	total number of pages requested.	Model Building
2	% Image	% of image pages (.gif/.jpg) requested.	Model Building
3	% Binary Doc	% of binary documents (.ps/.pdf) requested.	Model Building
4	% Binary Exec	% binary program files (.cgi/.exe) requested.	Model Building
5	robots.txt	indicates whether robots.txt file is requested	Class Labeling
6	% HTML	% of HTML pages requested.	Model Building
7	% Ascii	% of Ascii files (.txt/.c/.java) requested.	Model Building
8	% Zip	% of compressed files (.zip/.gz) requested.	Model Building
9	% Multimedia	% of multimedia files (.wav/.mpg) requested.	Model Building
10	% Other	% of other file formats requested.	Model Building
11	*totalTime*	approximated total time of the session.	Model Building
12	*avgTime*	average time between two successive HTML requests.	Model Building
13	*stdevTime*	standard deviation of time between successive requests	Model Building
14	*Night*	for HTML requests made between 12am to 7am.	Model Building
15	*Repeated*	fraction of repeated requests.	Model Building
16	*Error*	% of HTTP requests responded with status $\geq$ 400.	Model Building
17	GET	% of requests made with GET method.	Model Building
18	POST	% of requests made with POST method.	Model Building
19	HEAD	% of page requests made with HEAD method.	Class Labeling
20	OTHER	% of requests made with other methods.	Model Building
21	*width*	width of the traversal (in the URL space).	Model Building
22	*depth*	depth of the traversal (in the URL space).	Model Building
23	*length*	session length (total no of HTML requests).	Build data set
24	referrer = "-"	% of requests with empty referrer fields	Class Labeling
25	*MultiIP*	indicates whether session contains multiple IP	Model Building
24	*MultiAgent*	indicates whether session contains multiple agents	Model Building

at least two HTML requests. Sessions having more than two requests are again truncated by ignoring all the pages accessed after their second HTML request. This procedure is repeated until there are very few sessions left to create a new data set. (In our experiments, we have created data sets up to 7 requests).

D. Identifying Mislabeled Sessions

Mislabeled sessions in the training data can have adverse effects on the performance of the classification models. In this section, we describe a technique for identifying the mislabeled sessions using an ensemble filtering approach [11]. The idea here is to build multiple models from the training examples and use their misclassification errors to identify the mislabeled samples. In our approach, the multiple models are created from the data sets with different session lengths, and the misclassification errors are weighted according to the accuracy of each model. The C4.5 algorithm that we use prunes the decision trees it produces to avoid overfitting. As a consequence, the leaf nodes of the trees contain a probability distribution for each class label. We denote these probabilities as $P(0|X)$ and $P(1|X)$, where $P(i|X)$ is the probability that sample X belongs to class i.

Suppose there is an ensemble of k models, $C_1, C_2, \cdots, C_k$, built from the training data. Let $t(X)$ be the true class of sample X according to our session labeling scheme, while $C_m(X)$ is the class label assigned to X by the model C_m. Furthermore, let A_m denotes the accuracy of C_m. We define the false positive $FP_m(X)$ and false negative $FN_m(X)$ scores for each sample X and model C_m according to the following formula:

$$FP_m(X) = \begin{cases} 0 & \text{if } t(X) = C_m(X), \\ A_m \times |P(C_m(X)|X) - P(t(X)|X)| & \text{if } t(X) \neq C_m(X) \\ & \text{and } t(X) = 0. \end{cases} \quad (4)$$

$$FN_m(X) = \begin{cases} 0 & \text{if } t(X) = C_m(X), \\ A_m \times |P(C_m(X)|X) - P(t(X)|X)| & \text{if } t(X) \neq C_m(X) \\ & \text{and } t(X) = 1. \end{cases} \quad (5)$$

The overall false positive or false negative score for a sample X is given by $FP(X) = \sum_{m=1}^{k} FP_m(X)$ and $FN(X) = \sum_{m=1}^{k} FN_m(X)$. A large $FP(X)$ score indicates that the session X is currently assigned to be a non-robot despite being predicted to be a robot by most of the classification models, while a large $FN(X)$ score represents a potential non-robot session that is currently labeled as a robot.

E. Experimental Results

The experiments described in this section were performed on Web server logs recorded from January 1st until January 31st, 2001 at the University of Minnesota Computer Science department. The original Web log contains a total of 1,639,119 entries. After preprocessing, 180,602 sessions are created and the distribution of their robot and non-robot sessions are shown in Table V. The data set E0 (clean data) contains only sessions for which their user agents are known to be robots or non-robots, while E1 (noisy data) includes sessions with user agents that suggest them to be possible robot or non-robot sessions.

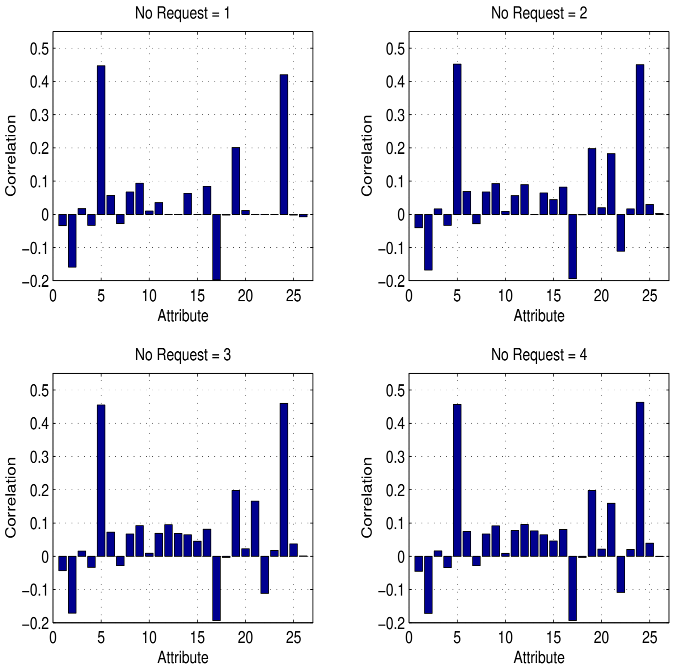

Fig. 6. Correlation between each feature and the class label at various number of requests. The horizontal axis corresponds to the feature Ids given in Table IV.

TABLE V

Distribution of robot and non-robot sessions at different number of HTML requests.

# HTML Request	E0 (Robot)	E0 (Non-robot)	E1 (Robot)	E1 (Non-robot)
1	8040	165835	10845	169757
2	3075	49439	3625	50640
3	2050	30277	2297	31080
4	1627	21436	1786	22075
5	1415	15618	1515	16141
6	1091	12108	1165	12532
7	908	9775	967	10154

Our initial task is to determine what are the features that correlate highly with Web robot sessions. Figure 6 shows the correlation of each feature with the class label as the session length increases from 1 to 4. As expected, the features used to determine the class labels (i.e. robots.txt, % HEAD and % unassigned referrer) have very strong positive correlation with the robot sessions. Although their correlations are relatively large, none of them exceeds 0.5. This suggests that using any of these features alone are insufficient to detect Web robots effectively. More importantly, the values of these features can be easily manipulated by robot designers to hide their true identities. For example, a robot designer can easily create a Web robot that does not use the HEAD request, does not access the robots.txt file, but assigns a non-empty referrer value to each of its request messages. The % image (feature 2) and % GET request (feature 17) have strong negative correlation with robot sessions, agreeing with our intuition that most robots tend to ignore image files while most browsers often use the GET request method to retrieve their documents. $avgTime$, $width$ and $depth$ are other good predictors for robot sessions after 2 requests. $width$ is positively correlated with robot sessions, whereas $depth$ is negatively correlated. This agrees with our previous example that shows many robot sessions have broad coverage but shallow depth unlike accesses by Web users (Figure 5).

E.1 Classifier Performance

For each data set in Table V, we randomly partition it so that their ratio of training to test sets are 7:3. Classification models are then built using the features for model building given in Table IV. We repeat the sampling and model building procedure 10 times for each data set. Figure 7 shows the average classification accuracy of the various models induced from our data sets. The results indicate that after four requests, we can obtain models with an average accuracy close to 90%. Also, their precision and recall (Figure 8) consistently stay above 82 % and 95% respectively, after more than three requests. The addition of noisy data (for E1) also does not degrade the classifier performance by much.

There is a dramatic improvement in all three performance measures when the number of requests increases from one to two. This is due to the fact that features such as $avgTime$, $width$, $depth$ and *Repeated* become meaningful after more than one request.

E.2 Finding Mislabeled Data

We used the technique described in Section III-D to identify sessions with high false positive and false negative scores. Figure 9 shows the log entries of the sessions with the highest false positive scores (i.e. sessions that are predicted to be robots but labeled as non-robots). We ob-

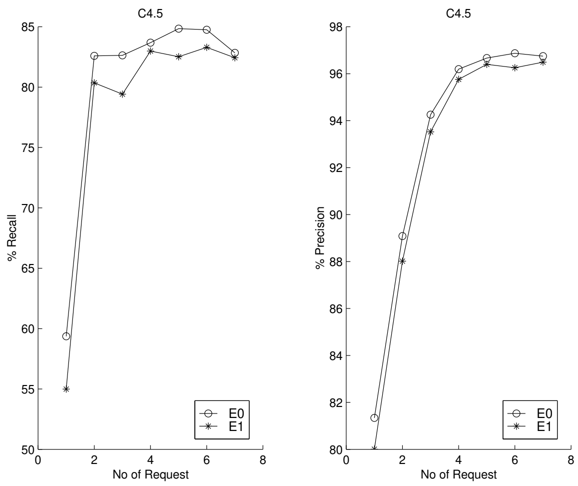

Fig. 8. Recall and Precision of classification models for different number of requests.

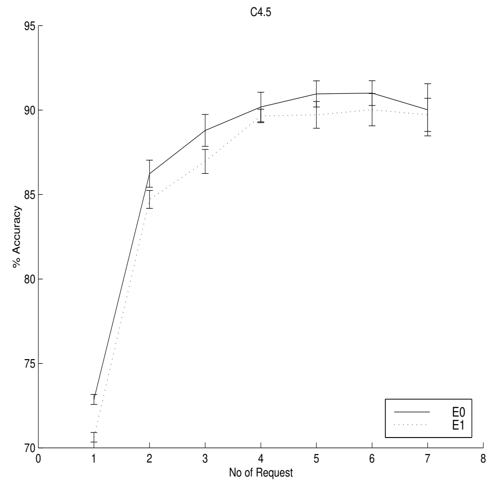

Fig. 7. Accuracies of classification models for different number of requests.

serve that these sessions are indeed being mislabeled due to camouflaging robots:

1. The first session contains a user agent that is very similar to a conventional browser. However, our models predict that it is a robot because it covers a significant portion of the Web site without focusing on any specific topic. Upon resolving the IP address of the session, our suspicion is confirmed because NEC Research, which owns the domain name, is known to have a Scientific Bibliography search engine.
2. The second example also looks highly suspicious despite having a user agent declared as Microsoft Internet Explorer. It is very unlikely that a user will be able to view all four HTML pages in one second. Unfortunately, we were unable to resolve their IP address to determine the origin of the client.
3. The third example is interesting because the pages retrieved by the session are mostly resume files. Our models were able to detect the large width of the traversal to infer that this session is most likely created by a robot. This observation is confirmed after resolving the IP address of the session (i.e. hire.com).
4. The fourth example is another session we believe is due to a resume hunter robot. However, it is interesting to note that the domain name of the client belongs to a broadband Internet Service Provider. Thus, the traditional approach of inspecting the user agent and IP address field of a Web client may not be able to reveal that it is a robot.

On the other hand, there are very few sessions labeled as robots but have high false negative scores (Figure 10). Only one of the four sessions shown in this figure is being mislabeled. `SilentSurf` (i.e. the fourth session) was initially thought to be the user agent of a robot. However, many of the classification models have predicted it to be a non-robot. Upon further examination, we found that `SilentSurf` is in fact an anonymizer Web site that changes the user agent of a conventional browser into a robot-like value. Therefore, it is clearly a mislabeled non-robot session. The rest of the false negative sessions in this figure are misclassification errors made by our models. These sessions are either robots whose navigational patterns are very similar to Web users (e.g. offline browsers) or download utility programs that have very short session lengths (which makes it difficult to predict their class labels correctly).

IV. Mining Issues

As previously noted, the current association mining framework has focused on discovering frequent patterns in the data, while ignoring those that do not satisfy the *minsup* threshold. As a result, one may end up removing useful patterns such as those involving negatively-correlated pages.

In this paper, we have applied a new data mining technique called indirect association [12] to Web click-stream data. Indirect association groups together association patterns that share a common set of items into more compact structures. It also points out the negative association that exist among items in the combined patterns. It turns out that from the Web Mining perspective, indirect association

IP Address/Hostname	Time	Requested Page	User Agent
kablam.nj.nec.com	14:47:23	/~jcui	Mozilla/4.05 [en] (Win95; U)
kablam.nj.nec.com	14:48:32	/~hsieh/misc/misc.html	Mozilla/4.05 [en] (Win95; U)
kablam.nj.nec.com	14:49:06	/Research/dmc/html/abstracts.html	Mozilla/4.05 [en] (Win95; U)
kablam.nj.nec.com	14:49:15	/~kencham/./abstracts/VCR-Ops.html	Mozilla/4.05 [en] (Win95; U)
kablam.nj.nec.com	15:14:03	/~gini/motion.html	Mozilla/4.05 [en] (Win95; U)
kablam.nj.nec.com	15:36:13	/~wijesek/research/qosMetrics.html	Mozilla/4.05 [en] (Win95; U)
64.3.57.99	5:06:42	/employment	Microsoft Internet Explorer/4.40.426 (Windows 95)
64.3.57.99	5:06:43	/grad-info	Microsoft Internet Explorer/4.40.426 (Windows 95)
64.3.57.99	5:06:43	/reg-info/csMinor.html	Microsoft Internet Explorer/4.40.426 (Windows 95)
64.3.57.99	5:06:43	/industry.html	Microsoft Internet Explorer/4.40.426 (Windows 95)
tpa1.hire.com	13:59:42	/~hngo/vnsa/may27-jul24/msg01844.html	Mozilla/4.75 [en] (X11; U; Linux 2.2.16-3 i686)
tpa1.hire.com	14:01:20	/~ssparikh/resume/shwetal_resume.html	Mozilla/4.75 [en] (X11; U; Linux 2.2.16-3 i686)
tpa1.hire.com	14:12:27	/~whalen/resume.html	Mozilla/4.75 [en] (X11; U; Linux 2.2.16-3 i686)
tpa1.hire.com	4:31:38	/~steinmet/pages/steinmetzresume.html	Mozilla/4.75 [en] (X11; U; Linux 2.2.16-3 i686)
rfx-64-6-194-38.users.reflexcom.com	14:51:00	/~myers/resume.html	Mozilla/4.0 (compatible; MSIE 5.01; Windows 98; DigExt)
rfx-64-6-194-38.users.reflexcom.com	14:58:25	/~tjiang/resume.html	Mozilla/4.0 (compatible; MSIE 5.01; Windows 98; DigExt)
rfx-64-6-194-38.users.reflexcom.com	15:03:45	/~littau/resume.html	Mozilla/4.0 (compatible; MSIE 5.01; Windows 98; DigExt)
rfx-64-6-194-38.users.reflexcom.com	15:11:17	/~tnnguyen/resume	Mozilla/4.0 (compatible; MSIE 5.01; Windows 98; DigExt)

Fig. 9. Sessions identified as having large false positive scores.

IP Address/Hostname	Time	Requested Page	User Agent
ns.mof.go.jp	18:42:16	/~subraman/cgi-bin/art.cgi	-
ns.mof.go.jp	18:48:13	/~subraman/cgi-bin/art.cgi	-
ns.mof.go.jp	18:48:20	/~subraman/arts/main.html	-
ns.mof.go.jp	18:48:20	/~subraman/arts	-
cip123.studcs.uni-sb.de	7:06:13	/~mobasher/webminer/survey/survey.html	Java1.1.8
cip123.studcs.uni-sb.de	7:20:51	/~mobasher/webminer/survey/survey.html	Java1.1.8
cip123.studcs.uni-sb.de	7:28:20	/~mobasher/webminer/survey/survey.html	Java1.1.8
212.160.138.34	8:16:31	/~hougen	Offline Explorer/1.3
212.160.138.34	8:21:05	/departmental	Offline Explorer/1.3
212.160.138.34	8:21:06	/Research/airvl	Offline Explorer/1.3
63.87.244.21	4:45:02	/~ssparikh	SilentSurf/1.1x [en] (X11; I; $MyVersion)
63.87.244.21	4:45:05	/~ssparikh/images/headsil.jpg	SilentSurf/1.1x [en] (X11; I; $MyVersion)
63.87.244.21	4:45:06	/~ssparikh/images/back/ivy.gif	SilentSurf/1.1x [en] (X11; I; $MyVersion)

Fig. 10. Sessions identified as having large false negative scores.

can be used to characterize the behavior of different groups of Web users who share a similar traversal path. We illustrate the idea of indirect association in the example below.

Example 2: Consider the example of Web sessions and site structure given in Figure 11. First, we find all the frequent itemsets in the Web data using the Apriori algorithm described previously. By imposing a minimum support threshold of 40%, any patterns that appear in less than 2 sessions are discarded. The support of size-2 patterns and size-3 frequent patterns are shown in Figure 11. We would discover that the page D appears frequently with both A and B in the same session. Similarly, E also appears frequently with both A and B. Without any prior knowledge about the structure and content of the pages, we would expect D and E to be visited together quite often. However, since the joint support of D and E is below the minimum threshold (and the pages are negatively correlated), we say that they are indirectly associated via the *mediator* set $\{A, B\}$. We can also show that an indirect association exists between A and C via $\{B, D\}$.

Indirect association cannot be easily found using the standard association mining framework because one has to examine both the positive and negative dependencies

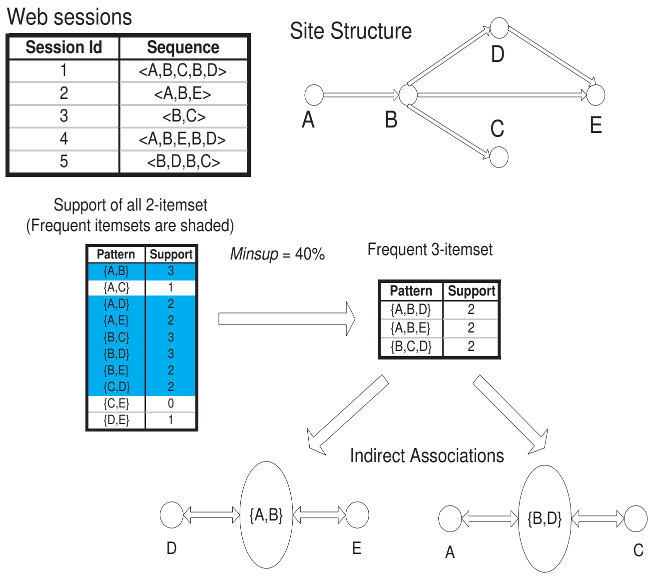

Fig. 11. Example of Indirect Association for Web data.

between items. Although there have been attempts to mine both positive and negative associations simultaneously, such algorithms can be prohibitively expensive. Furthermore, the information conveyed by these patterns are quite different from that of indirect association. For instance, a mixed itemset $\{A, B, \overline{C}\}$ has a different meaning than an indirect association between A and C via B.

The indirect association formulation also provides a methodology to address the impact of Web site structure. By grouping together patterns that share a common set of navigation pages, we can reduce the number of patterns presented to an analyst while retaining the information contained in each of the pattern. In most cases, the navigation pages would appear as mediators for various indirect associations. In other cases, the navigation pages themselves could be indirectly associated, which is interesting because they could reveal the interest for different groups of Web users.

A. Indirect Association: Formulation

Definition 1: A pair of items, a and b, is said to be indirectly associated via a mediator set M if the following conditions hold :

1. $sup(\{a,b\}) < t_s$ (Itempair Support condition)
2. There exists a non-empty set M such that :
 (a) $sup(\{a\} \cup M) \geq t_f, sup(\{b\} \cup M) \geq t_f$ (Mediator Support condition).
 (b) $d(\{a\}, M) \geq t_d,\ d(\{b\}, M) \geq t_d$ where $d(P,Q)$ is a measure of the dependence between the itemsets P and Q (Mediator Dependence condition).

Condition 1 is needed because an indirect association between a and b is meaningful only if both items rarely occur together in the same transaction. Otherwise, it makes more sense to characterize the pair in terms of their direct association. Alternatively, condition 1 can be modified to test for independence between items a and b. However, it is often the case that itempairs that have very low support values are either independent or negatively correlated with each other [13]. Thus, it is often sufficient to use condition 1 to discover indirect association between independent or negatively correlated itempairs.

Condition 2(a) can be used to ensure that the mediator M is statistically significant as an intermediary between a and b. From a computational perspective, the support measure also has a nice anti-monotonicity property that allows us to reduce the exponential number of candidate mediators.

Condition 2(b) ensures that only itemsets that are highly dependent on both a and b are used to form the mediator M. For instance, suppose there is an item k that appears in every transaction. Without the mediator dependence condition, any itempairs that are infrequent will be indirectly associated, by default, via k. Thus, condition 2(b) is necessary to remove indirect associations due to such uninformative mediators. There are many interest measures we can be use to represent the degree of dependencies among attributes of a dataset [14]. One such measure is the ϕ coefficient, which is analogous to Pearson's product-moment correlation coefficient for continuous variables. For pairs of binary attributes, it can be shown that within certain range of support values [4], the correlation between X and Y, $\phi_{X,Y}$, can be expressed in terms of their interest factor, $I(X,Y) \equiv \frac{P(X,Y)}{P(X)P(Y)}$ [15], [16], and their joint probability (support), i.e. :

$$\phi_{X,Y} \approx \sqrt{\mathrm{I(X,Y) \times P(X,Y)}} \equiv IS(X,Y)$$

We will use the right-hand side of the above expression, called the IS measure, to be the dependence measure for Condition 2(b) [13]. This measure is desirable because it takes into account both the statistical dependence and statistical significance of a pattern. We do not use $\phi_{X,Y}$ as the dependence measure because it treats both the co-presence and co-absence of items in the same way. For many application domains involving sparse data sets such as the Web click-stream data, co-presence of items is more important than co-absence.

Nonetheless, our indirect association formulation can accommodate other interest measures, such as Piatetsky-Shapiro's rule-interest, J-measure, mutual information and Gini index. This formulation can also be extended to a more general case where a and b are itemsets rather than individual items.

An algorithm for mining indirect association between pairs of items is shown in Table VI. There are two phases in this algorithm: (1) Candidate generation, and (2) Candidate pruning. During the first phase, we use a standard frequent itemset generation algorithm such as Apriori [3] to generate the frequent itemsets. Next, we combine the frequent itemsets of size-k (L_k) to generate candidate indirect associations for pass $k+1$ (C_{k+1}). Each candidate in C_{k+1} is a triplet, $< a, b, M >$, where a and b are the indirect pairs, associated via the mediator M. During the join step (Step 4) of this algorithm, a pair of frequent k-itemsets, $\{a_1, a_2, \cdots, a_k\}$ and $\{b_1, b_2, \cdots, b_k\}$, is combined together to produce a candidate indirect association $< a, b, M >$ if the two itemsets have exactly $k-1$ items in common. Since C_{k+1} is generated from frequent itemsets, the mediator support condition is trivially satisfied. The pruning phase (steps 5 through 7) is used to eliminate candidates that fail the itempair support and mediator dependence conditions (see Definition 1). One way to check the support of each itempair is by creating an $M \times M$ support matrix for all pairs of frequent items, where $|L_1| = M$. Thus, the itempair support condition can be easily verified using a matrix lookup operation. Alternatively, we may store the candidate indirect itempairs in a hash tree and performs an additional pass over the data set in order to count the actual support of each candidate itempair. The latter approach can be used to handle the case where a and b are itemsets rather than individual items.

We now briefly describe the complexity of the above algorithm. The candidate generation phase can be quite expensive, because it requires at most $O(\sum_k |L_k| \times |L_k|)$ join op-

[4] when $P(X) \ll 1$, $P(Y) \ll 1$ and $\frac{P(X,Y)}{P(X)P(Y)} \gg 1$.

TABLE VI

Algorithm for mining indirect association between pairs of items.

1. Extract the large itemsets, $L_1, L_2, \cdots L_n$, using standard mining algorithms.
2. $P = \emptyset$
3. for $k = 2$ to n do
4. $\quad C_{k+1} \leftarrow \text{join}(L_k, L_k)$
5. $\quad$ for each $(a, b, M) \in C_{k+1}$ do
6. $\quad\quad$ if $(sup(\{a, b\}) < t_s$ and $d(\{a\}, M) \geq t_d$ and $d(\{b\}, M) \geq t_d$
7. $\quad\quad\quad P = P \cup (a, b, M)$
8. $\quad$ end
9. end

erations. Nonetheless, we have implemented several optimization strategies to reduce the number of join operations. For example, given a frequent itemset $\{a_1, a_2, \cdots, a_k\}$, we only need to perform the join operation with other frequent itemsets $\{b_1, b_2, \cdots, b_k\}$ for which the condition $a_2 \geq b_1$ is satisfied. This condition holds because all items within an itemset are kept in lexicographic order. The candidate pruning phase is inexpensive if the $N \times N$ itempair support matrix can fit into main memory. Otherwise, the complexity of this step can be as expensive as the candidate counting step of the Apriori algorithm.

We can also combine the various indirect associations that share a common mediator. For example, suppose (a, b) and (a, c) are indirectly associated via the mediator M. We can construct a graph $G = (V, E)$ where $V = \{a, b, c\} \cup M$ and E is the edges between the vertices in V. Each edge $e = (v_i, v_j) \in E$ indicates whether a direct or indirect association exists between the two connected vertices. This association graph is a compact representation of dependence relationships among all items (itemsets) that share a common mediator. We have implemented an indirect association viewer to visualize the graphs for each mediator. Unlike other association pattern viewers [17], our technique uses both the support and dependence measures to analyze the interaction between various items. Furthermore, instead of building a global association graph for all items, our viewer is mediator-centric.

B. Experimental Results

To demonstrate the utility of indirect associations, we have performed our experiments on Web server logs from an e-commerce organization. The data set contains 6664 pages and 143,604 sequences. The thresholds used for our experiments are $t_s = 0.005$, $t_f = 0.05$ and $t_d = 0.2$. We observe many interesting indirect associations among different groups of Web users. Figure 12 illustrates one such example. The solid lines indicate that the support between the two vertices is high while the dashed lines indicate that there is an indirect association between the two vertices. The mediator set corresponds to the itemset {Home page, jewelry, diamond}. This itemset corresponds to the pages that are often visited by users who are looking for jewelry items. Note that the men's jewelry page is indirectly associated with other jewelry pages, such as the anniversary page and the ladies page. This suggests that visitors who are shopping for male jewelry are seldom interested in other jewelry pages.

Another example is given in Figure 13, which shows the different groups of users who view the video Web page. This include users who are only interested in camcorders; those who are interested in video games and entertainment pages; those who are looking for televisions; and finally, Web users who are looking for VCRs. It is somewhat surprising to find that users who access the television pages are not interested in the VCR pages, and vice-versa.

V. Conclusions

In this paper, we have described the various challenges involved in mining association patterns from Web data. First, we address the robot detection issue by building classification models using access features derived from the Web server logs. The results of our experiments on the Computer Science department Web server logs suggest that Web robots can be detected with more than 90% accuracy after 4 requests. These classification models can also be used to detect robots that attempt to disguise their identities. Second, we have applied a new data mining technique called indirect association to Web click-streams. This technique combines similar association patterns into a more compact structure and looks for negatively correlated subpatterns within the combined structure. This pattern is useful to highlight the differing interests of Web users who share a common traversal path.

Acknowledgments

This work was partially supported by NSF grant # ACI−9982274 and by Army High Performance Computing Research Center contract number DAAD19−01−2−0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by AHPCRC and the Minnesota Supercomputing Institute.

References

[1] J. Srivastava, R. Cooley, M. Deshpande, and P.N. Tan, "Web usage mining: Discovery and applications of usage patterns from web data," *SIGKDD Explorations*, vol. 1, no. 2, pp. 12–23, 2000.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: a performance perspective," *IEEE Transactions on Knowledge and Data Eng.*, vol. 5, pp. 914–925, 1993.

[3] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. of the 20th VLDB Conference*, Santiago, Chile, 1994.

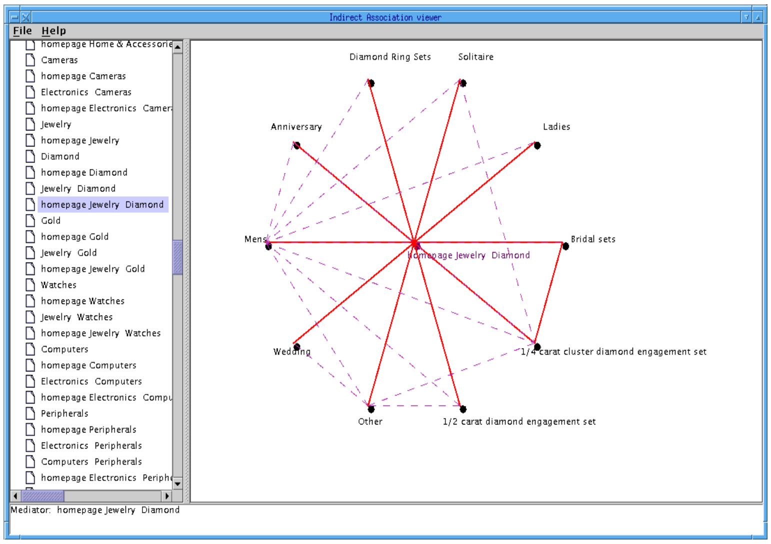

Fig. 12. Indirect association among visitors who are interested in the jewelry and diamond Web pages.

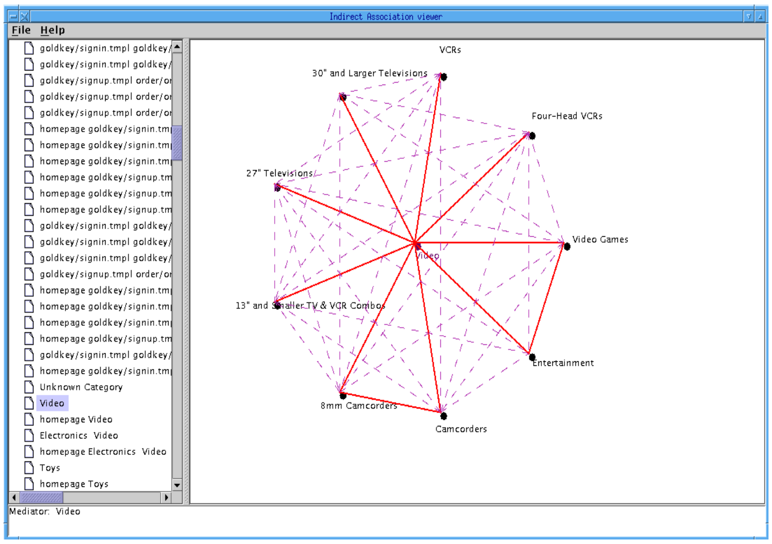

Fig. 13. Indirect association among visitors who are interested in the video Web page.

[4] P.N. Tan and V. Kumar, "Discovery of web robot sessions based on their navigational patterns," *accepted for the special issue of the International Journal of Data Mining and Knowledge Discovery on Web Mining for E-commerce*, 2001.

[5] L. Graham, "Keep your bots to yourself," *IEEE Software*, vol. 17, no. 6, pp. 106–107, November/December 2000.

[6] M. Yoon, "Web robot detection," http://www.arsdigita.com/doc/robot-detection, 2000.

[7] S. Jackson, "Building a better spider trap," http://www.spiderhunter.com/spidertrap, 1998.

[8] M. Koster, "A standard for robot exclusion," http://info.webcrawler.com/mak/projects/robots/norobots.html, 1994.

[9] C.P. Kolar, J.R.R. Leavitt, and M. Mauldin, "Robot exclusion standard revisited," http://www.kollar.com/robots.html, 1996.

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

[11] C. Brodley and M.A. Friedl, "Identifying mislabeled training data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[12] P.N. Tan, V. Kumar, and J. Srivastava, "Indirect association: Mining higher order dependencies in data," in *Proc. of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Lyon, France, 2000, pp. 632–637.

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

[14] P.N. Tan, V. Kumar, and J. Srivastava, "Selecting the right metric for data mining application," Tech. Rep., University of Minnesota, 2001.

[15] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Proc. of 1997 ACM-SIGMOD Int. Conf. on Management of Data*, Tucson, AZ, 1997.

[16] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, "Using association rules for product assortment decisions : A case study," in *Proc. of the Fifth Int'l Conference on Knowledge Discovery and Data Mining*, San Diego, August 1999.

[17] A. Wexelblat, "An environment for aiding information-browsing tasks," in *Proc. of AAAI Symposium on Acquisition, Learning and Demonstration: Automating Tasks for Users*, Birmingham, UK, 1996.