Modeling of Web Robot Navigational Patterns

Pang-Ning Tan[1] and Vipin Kumar[1]

Department of Computer Science,
University of Minnesota,
200 Union Street SE,
Minneapolis, MN 55455.
{ptan,kumar}@cs.umn.edu

Abstract. In recent years, it is becoming increasingly difficult to ignore the impact of Web robots on both commercial and institutional Web sites. Not only do Web robots consume valuable bandwidth and Web server resources, they are also making it more difficult to apply Web Mining techniques effectively on the Web logs. E-commerce Web sites are also concern about unauthorized deployment of shopbots for the purpose of gathering business intelligence at their Web sites. Ethical robots can be easily detected because they tend to follow most of the guidelines proposed for robot designers. On the other hand, unethical robots are more difficult to identify since they tend to camouflage their entries in the Web logs. In this paper, we examine the problem of identifying navigational patterns of Web robots using conventional machine learning techniques. Our goal is to construct a predictive model that will distinguish between the browsing behavior of legitimate Web users from access patterns due to Web robots. Our results show that highly accurate models can be obtained using a small set of access features deduced from the Web logs.

1 Introduction

Ever since Web robot [1] first appeared in 1993, the ease with which it can be constructed has caused a rapid proliferation of various types of Web agents on the Internet. A Web robot is an autonomous agent that traverses the hyperlink structure of the Web for the purpose of locating and retrieving information from the Internet. The first Web robot was developed to discover and count the number of Web servers. Nowadays, Web robots are often used as resource discovery and retrieval tools employed by Web search engines [1, 6], shopping comparison robots [3], email collectors [4, 2], offline browsers [8, 7], browsing assistants [11, 20], etc. They are also used by Web administrators for site maintenance purposes (such as mirroring and checking for dead hyperlinks). Furthermore, various programming languages, such as Perl and Python, have built-in library functions to facilitate the construction of agents with Web crawling abilities. All these have created a Web environment in which one can no longer ignore the impact of Web robot visits to a particular Web site.

There are a number of situations in which it is desirable to identify Web robots and distinguish them from other accesses. First of all, e-commerce Web sites may be concerned about unauthorized deployment of shopbots for gathering business intelligence at their Web site. In such cases, it will be desirable to disable accesses by non-authorized robots. Secondly, visits by Web robots can distort the access patterns that are mined by Web Usage Mining systems. For example, at least 10% of the server sessions in the University of Minnesota Computer Science department Web logs can be attributed to Web robots. Many of these robots adopt a breadth-first retrieval strategy to increase their coverage of the Web site. Due to such accesses, the association rule mining system may inadvertantly generate frequent itemsets involving Web pages from multiple categories. Such spurious patterns may lead the analyst of an e-commerce site to believe that Web users are interested in products from the different categories when in fact such patterns are induced by robot sessions. This can be avoided if we are able to remove Web robot sessions from the dataset during preprocessing. Another reason is because the deployment of Web robots usually comes at the expense of other users, as they may consume considerable network bandwidth and server resources. Occasionally, poorly-designed robots can be trapped in an infinite loop of the URL space. It will be desirable to detect such behavior and disable them as soon as possible.

[1] also known as Web crawler, spider or worm.

In this paper, we describe the problem of identifying Web robots based on their access features deduced from the Web logs. Our goal is to construct predictive models that will distinguish between robots and non-robots navigational patterns. In addition, we have also attempted to determine the minimum number of clicks required to identify the owner of a session, whether it is a robot or non-robot, with reasonable accuracy.

The rest of the paper is organized in the following way. In Sect. 2, we formalize the problem of Web robot detection and discuss the different types of Web robots available today. Section 3 describes the various features used for building the predictive models. This is followed by our experimental results in Sect. 4. Finally, Sect. 5 presents our concluding remarks with suggestions for future research.

2 Web Robots : Overview

2.1 Types of Web Robots

In order to understand the navigational patterns of Web robots, it is important to know the different types of Web robots that are available today. This is because different types of robots may exhibit different access patterns. Such knowledge can be used to identify the set of relevant features that best characterize the access pattern of a Web robot.

Eichman [13] divides Web robots into two distinct categories : agents that are designed to accomplish a specific task (such as browsing assistants or hyperlink checkers) and agents that are used to build information bases (e.g. search engine robots such as T-Rex for Lycos [6] and Scooter for Altavista [1]). Search engine robots often use breadth-first search to maximize their coverage of a Web site.

Browsing assistants are interactive agents that can help Web users to locate relevant documents on the Web by recommending hyperlinks based on the current user profile [11, 20]. These agents often rely on an underlying Web spidering mechanism to retrieve pages, and a learning module to predict the relevance of these pages. Unlike the general-purpose search engines, browsing assistant robots adopt a more direct search strategy focussing on reference links that will lead them to documents of interest.

Hyperlink checkers are Internet utility programs used by Web site administrators to test for broken links or missing pages [9, 5]. Many of these utilities use the HEAD request method to test for the validity of a link.

Kephart et al.[14] defines shopbots as programs that automatically search for information regarding the price and quality of goods or services on behalf of a consumer. The deployment of shopbots has received mixed reactions from different vendors. Some vendors attempt to block accesses by these robots while others perceive them as a means of attracting potential customers.

Email collectors [4, 2] are robots that automatically collect email addresses posted on the Web. The search strategy employed by such systems is often directed towards discovering personal home pages rather than e-commerce Web sites.

Offline browsers are either stand-alone browsers or add-on utilities that allow a Web user to download an entire Web site to the local directory for offline viewing [8, 7]. In this paper, we will treat such browsers as similar to Web robots, primarily due to their spidering capabilities.

2.2 Guidelines for Web Robots

Eichman [13] and Koster [17, 15] have provided several ethical guidelines for Web robot developers. The purpose of these guidelines is to ensure that both the Web robot and Web server can cooperate with each other in a way that will benefit both parties. Under these guidelines, a Web robot must identify itself to the Web server and moderate its rate of information acquisition. The Robot Exclusion Standard [16, 10] was proposed to allow Web administrators to specify which part of their site are off-limits to visiting robots. Robots that follow the proposed guidelines can be easily identified according to the following characteristics :

1. An ethical robot always access the robots.txt file before downloading other documents from the Web site.
2. An ethical robot uses the User-agent and From fields of the HTTP request header to declare its identity to the Web server.
3. An ethical robot helps to minimize the burden on Web servers by using a low retrieval rate and the HEAD request method, whenever possible, or by executing only when the server is lightly loaded.

Unfortunately, not every Web robots obey these noble guidelines. This has created a need to find new ways of identifying such robots. These robots often try to conceal their identity in several ways :

1. by ignoring the robots.txt file.
2. by creating several concurrent HTTP sessions with a Web server; each having a different IP address. In some cases, the robots.txt file is

retrieved by one of these sessions. If robot detection is based on access to this file alone, the rest of the robot sessions may go undetected.
3. by using multiple agent information.
4. by using the same agent information as conventional Web browsers.

2.3 Robot Detection Problem

Despite the various attempts to regulate the behavior of Web robots, not all of them are successful. This is because implementation of such guidelines require full cooperation from the Web robot designer. Some robots may want to remain anonymous while traversing a Web site. Therefore, other means of robot identification are necessary. In this paper, we will investigate how machine learning techniques can be applied to Web server logs for automatic detection of Web robots.

Our goal of is to construct a classification model capable of detecting the presence of Web robot sessions in a fast and accurate manner. Accuracy is a crucial requirement. However, the cost of misclassifying user sessions may far outweigh that of robot sessions. This is where cost-sensitive learning techniques can be applicable. Speed is another important issue. The classification model should rely on features that can be derived easily and quickly from the Web logs. Furthermore, early detection will be advantageous to prevent more resources being tied up by uncooperative Web robots.

3 Methodology

3.1 Data Source and Preprocessing

In this paper, we will adopt the Web characterization terminologies defined by the WWW Consortium (W3C) [19]. These terminologies include the definitions of clients, users, clicks, server sessions, temporal session length, server path length, etc (see Table 1).

The Web server log is the data source used in our experiments. A typical Web log contains information such as the IP address of the client, the date and time a request is made, the method and protocol used, the URI of the requested page, the status of the request handling, the size of the document retrieved, the referrer page and the client agent information.

During preprocessing, the log entries are grouped into server sessions according to the IP Address and agent information of the client [21, 12]. New sessions are also created using a 30-minute timeout period. For each

Table 1. Some of the Web characterization terminologies defined by W3C [19].

Client	an application which is capable of accessing Web resources; e.g, Web browser or robot.
User	a human using a client to *interactively* retrieve Web resources.
Request	an atomic operation to be carried out in the context of a specified resource.
Click	a user-initiated request to dereference a Web resource.
Embedded click	a request for dereferencing a URI that is embedded within a Web resource; e.g., by clicking on a hyperlink.
User-input click	a request for dereferencing a URI supplied directly by the user to the Web client; e.g., by typing into the address window, or using bookmarks, history profile etc.
User session	a delimited set of user clicks across one or more Web servers.
Server session	a collection of user clicks to a Web server during a user session.
Page View	visual rendering of a Web page in a specific client environment at a specific point in time.
Temporal session length	the amount of time elapses during the course of a user session.
Server path length	number of clicks to a Web server during a user session.

session, we have identified the log entries associated with each click. One way to aggregate the log entries into clicks or page-views is by parsing the entire site files for embedded links to be associated with each HTML page. An alternative approach is to approximate the duration of each click using the timestamp, page type and referrer fields of the log entries. For example, if the referrer field between two successive requests are different, then they could be associated with two separate clicks. Otherwise, if the referrer field is the same (but not equal to "-"), and the time difference is less than 2 seconds, then we may infer that they belong to the same user click. Otherwise, they would be associated with different click events. The type of requested page also play an important role in distinguishing user clicks because a page view quite often involves a set of image files associated with an HTML page. For example, if the requests involve two successive image pages (within a duration of 5 seconds), both having a "-" referrer field, it is very likely that they both belong to the same page view.

3.2 Feature Selection

Once the server log entries have been sessionized, the next step is to construct a feature vector representation for each session. Table 2 presents a summary of the key attributes derived from the server sessions. Some of these attributes are obtained directly from the Web logs while others are computed from the session information. For example, robots.txt is an attribute that can be directly inferred from the server logs. On the other hand, totalPages, % Image, % Binary Doc, etc., are obtained by counting the different types of Web pages requested for a particular session. The computation of temporal attributes such as totalTime, AvgTime and stdevTime is illustrated in Fig. 1. The totalTime attribute can be approximated by the interval between the timestamp of the last log entry for the previous click and the timestamp of the first log entry of the server session. On the other hand, avgTime and stdevTime is computed using the intervals between successive clicks of the session. It is intuitively clear that both attributes are not computable using information after one click. The width and depth attributes are computed by constructing a representative graph based on the URI names of the requested pages. For example, if a session contains requests for the following pages, {/A, /A/B, /A/B/C}, then its width will be 1 and its depth will be 3. Basically, the width attribute measures the number of leaf nodes generated in the graph while the depth attribute measures the maximum depth of the tree(s) within the graph. Therefore, a session that contains requests for {/A, /A/B, /C, /D} will have a width of 3 and a depth of 2.

Table 2. Summary of attributes derived from server sessions. The attributes are used for class labeling (denoted as Classify) or constructing the feature vector representation (Feature).

Id	Attribute Name	Remark	Purpose
1	totalPages	Total number of pages requested.	Feature
2	% Image	% of image pages (.gif/.jpg) requested.	Feature
3	% Binary Doc	% of binary documents (.ps/.pdf) requested.	Feature
4	% Binary Exec	% binary program files (.cgi/.exe/.class) requested.	Feature
5	robots.txt	No. of times robots.txt is accessed.	Classify
6	% HTML	% of HTML pages requested.	Feature
7	% Ascii	% of Ascii files (.txt/.c/.java) requested.	Feature
8	% Zip	% of compressed files (.zip/.gz) requested.	Feature
9	% Multimedia	% of multimedia files (.wav/.mpg) requested.	Feature
10	% Other	% of other file formats requested.	Feature
11	totalTime	Temporal server session length (approx.).	Feature
12	Avg time	Average time between clicks (approx.).	Feature
13	Stdev time	Standard deviation of time between clicks (approx.).	Feature
14	Night	% of requests made between 2am to 6am (local time).	Feature
15	Repeated	Reoccurence rate of file requests.	Feature
16	Error	% of requests with status $\geq$ 400.	Feature
17	GET	% of requests made with GET method.	Feature
18	POST	% of requests made with POST method.	Feature
19	HEAD	% of page requests made with HEAD method.	Classify
20	OTHER	% of requests made with other methods.	Feature
21	width	width of the traversal (in the URL space).	Feature
22	depth	depth of the traversal (in the URL space).	Feature
23	pathLength	Server path length (no of clicks).	Feature
24	referrer = “-”	% of requests with referrer = “-”	Classify

Some of the attributes in Table 2 will be used to assign the appropriate class label of a session while others are used in the feature vector construction. The assignment of class labels can be done in the following way :

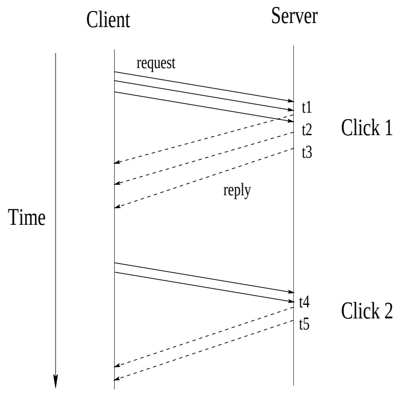

Fig. 1. This session contains two user clicks. $t1$, $t2$, $t3$, $t4$ and $t5$ are the timestamps recorded in the server logs. The approximate temporal session length is $t5 - t1$ while the period between the two clicks is $t4 - t3$.

1. If a session contains a request for the robots.txt file, then the session is identified as a robot session (denoted as Class = 1).
2. If the agent field of a session corresponds to the agent of a known Web robot, then Class = 1. An initial list of agent information for known Web robots is needed.

There are other heuristics we can use to supplement the above labeling scheme. For example, if all the requests are made using the HEAD method, then the session is most likely created by a link checker robot or a proxy server. In this paper, sessions created due to cache validation requests by proxy servers will be classified as non-robots. Another heuristic could be based on the referrer field of the session. If all the requests by a Web client use "-" as the referrer field, then there is a strong possibility that the client is a Web robot, as long as the number of clicks is large. If number of clicks is small, the session is more likely created by a Web user. This is because a Web browser always use the "-" value as its referrer field during a user-input click. For longer sessions, the probability that a Web browser generates only user-input clicks are much lower. By selecting an appropriate threshold on the minimum number of clicks, one can potentially identify new robot sessions. Later, we will show that the number of Web robots generated this way is quite small compare to the one generated by other means. Thus, the accuracy of the models will not be affected by much if we ignore the two heuristics.

Another important task is to identify the set of relevant features to represent each session. There are two main criteria for choosing the appropriate feature : (1) It should be easily derived from the Web logs, and (2) It should not be easily manipulated by robot designers who wish to conceal their identity. This rules out the obvious attributes such as robots.txt.

Figure 2 illustrates the correlation [2] between each attribute with the class label. The bar graph is plotted for the various number of clicks within a session. The following results are observed :

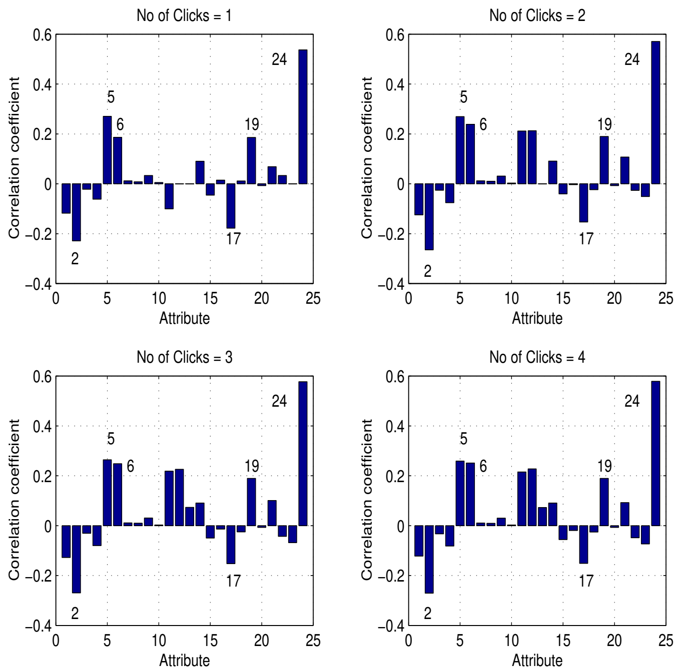

Fig. 2. Correlation between access attributes and the Robot class label for various session path lengths. The labels accompanying some of the vertical bars indicate the attribute ids.

1. As expected, the classification attributes (i.e. attributes 5, 19 and 24) have very strong positive correlation with the class labels. These attributes can be used to detect ethical robot sessions but can also be easily manipulated by other robot designers.

[2] Note that linear correlation may not be the best measure of attribute dependence when non-linear dependencies exist in the data.

2. After two clicks, both the totalTime and AvgTime attributes become more significant. The % HTML attribute also becomes increasingly important.
3. At the end of three clicks, the correlation plot looks quite similar as before, except that the standard deviation attribute becomes more important (standard deviation is always zero prior to three clicks).
4. The width and depth attributes have opposite effects on the class label. Figure 2 show that robot sessions tend to have broader width and smaller depth, indicative of a breadth-first retrieval strategy.

3.3 Classification

Once the set of relevant features have been identified, classification models are built using the standard C4.5 algorithm. As mentioned earlier, there are two main objectives we would like to achieve : (1) to find a good predictive model for Web robot access patterns, and (2) to find the minimum number of clicks required to produce reasonably accurate models.

Accuracy is not the only metric we use to evaluate the performance of our classifiers. In the area of information retrieval, recall and precision are two popular metrics used to evaluate binary classifiers :

$$\text{recall}, r = \frac{\text{no of robot sessions found correctly}}{\text{total no of actual robot sessions}} \tag{1}$$

$$\text{precision}, p = \frac{\text{no of robot sessions found correctly}}{\text{total no of predicted robot sessions}} \quad . \tag{2}$$

A classifier that assigns the value 1 to every session will have perfect recall but poor precision. In practice, the two metrics are often summarized into a single value, called the F_1-measure [22] :

$$F_1 = \frac{2rp}{(r+p)} \quad . \tag{3}$$

This value is maximized when r and p are close to each other. Otherwise, the value of F_1-measure is dominated by the smaller of r and p [23].

4 Empirical Results

Our experiments were performed on the University of Minnesota Computer Science department server logs collected from April 10th to April 24th, 2000. Initially, the Web log contains 330,401 log entries. After preprocessing, 47925 sessions are created; out of which 5442 sessions are labeled as robot sessions while the rest are non-robot sessions. There are only 172 non-robot sessions with HEAD = 1, which constitutes less than 0.4% of the overall dataset. Furthermore, only 36 of these sections have more than 1 click. Therefore, even though our dataset is not labeled based on the HEAD method, we believe that the error of mislabeling sessions with HEAD requests is quite negligible. There are also 5253 robot sessions and 6304 non-robot sessions with referrer = "-". Among the non-robot sessions, 4824 of them have only a single click and 79 of them have more than 10 clicks. Since the referrer = "-" heuristic is useful only when the number of clicks are large, we believe that accuracy of our models will not be affected if we ignore this heuristic.

Each session is then converted into a feature vector representation. Some of the sessions contain a single click, while others may have more than one click. We then create a dataset for each click size. For instance, the dataset for one click is generated from all sessions that contain at least one user click. Sessions with more than one click will be truncated by computing the feature values up to their first click. For dataset with two clicks, we ignore all single click sessions, and consider only sessions with at least two clicks while truncating sessions of larger click size. A summary of the dataset parameters for the various number of clicks is given in Table 3. For each dataset, we partitioned the samples into training and test sets according to the ratio of 3:2. The training set is created via random sampling on the full dataset. We also ensure that there will be an equal number of robot and non-robot sessions in the training set. To evaluate the accuracy of our classifiers, we generate a test set that contains the same proportion of robot and non-robot sessions. For recall and precision calculations, the test set is comprised of all the sessions not included in the training set. The C4.5 algorithm is used to build the classification model for each dataset. Our sampling and model building procedure is repeated 20 times for each dataset.

Figure 3 illustrates the overall classification accuracies for various models induced from the dataset. Our results show that after two clicks, the accuracy improves from 71% to almost 90%. The main reason for such a dramatic improvement is due to the addition of totalTime and avgTime attributes that become available after 2 or more clicks. Also, our precision and recall results consistently reach above 89 % after more than one click. The improved precision at two clicks indicates that the addition of avgTime and totalTime attributes reduce the amount of false positives due to misclassification of non-robots as robots. Our overall accuracy would continue to improve gradually as the number of clicks increases until it

Table 3. Summary of dataset parameters and results of experiment.

Number of Clicks	No of robot Sessions	No of non-robot Sessions	Average Accuracy (%)	Average Precision	Average Recall	F_1 Measure
1	5442	42483	71.52	0.6880	0.9571	0.8005
2	1872	17986	89.54	0.8998	0.9033	0.9015
3	849	10956	89.94	0.9139	0.8984	0.9061
4	526	7626	89.74	0.9276	0.8972	0.9121
5	384	5643	90.52	0.9188	0.8997	0.9091

begins to deteriorate after 5 clicks. This can be explained by the fact that the sample size decreases significantly when the number of clicks are larger than 5. The graph on the right side of Fig. 3 illustrates how the average training and test error changes with sample size. When the number of clicks increases, the training error continues to decrease. However, the curve for the test error levels off. This suggests that the sample size may not be large enough to build a good predictive model. Improvement in the training error for large click size can be attributed to overfitting of the dataset; it does not add any predictive power on the test set.

The results of C4.5 decision tree algorithm will be used to generate classification rules by using the C4.5rules algorithm. Our discussion will be based on the output of C4.5rules since classification rules are easier to interpret. Table 4 presents some of the classification rules obtained for the various datasets. Below, we highlight some of the rules found (in one or several models generated for each dataset) :

1. With a single click, the rules that characterize non-robots vary from one set of samples to another. This explains the poor precision of the results. One of the most consistent rule found is % Binary Doc $>$ 0.6666, Night ≤ 0, GET $> 0.5 \longrightarrow$ class 0; i.e., sessions that use the GET request to access a significantly large number of binary documents during the day are most likely non-robots. The confidence of this rule is between 70$-$75% while its support is quite low. On the other hand, robot sessions can be characterized by the absence of requests for image files. For example, the second rule in Table 4 indicates that if a session occurs in the middle of night and less than half of the pages retrieved are image files, then there is a strong likelihood that the session belongs to a Web robot.
2. After two clicks, the totalTime attribute helps to improve the accuracy of predicting non-robot sessions. For example, the first rule for 2 clicks

Table 4. Some of the decision rules produced by various classification models using different number of user clicks.

No of Clicks	Induced rules
1	% Binary Doc > 0.6666, Night ≤ 0, GET $> 0.5 \longrightarrow$ class 0 $0 <$ % Image ≤ 0.4615, Night $> 0 \longrightarrow$ class 1
2	% Image > 0, totalTime $\leq 594 \longrightarrow$ class 0 $59 <$ totalTime ≤ 121, Repeated $> 0.4615 \longrightarrow$ class 0 % Image ≤ 0, Error Request ≤ 0, width $> 3 \longrightarrow$ class 1 totalPages > 10, % Images $\leq 0 \longrightarrow$ class 1 Error Request $> 0 \longrightarrow$ class 1
3	total Pages > 3, % Image $> 0.3636 \longrightarrow$ class 0 % Binary Exec $> 0.4444 \longrightarrow$ class 0 % Image ≤ 0, totalTime > 1016, stdevTime $\leq 391.03 \longrightarrow$ class 1 % Image ≤ 0, width $> 4 \longrightarrow$ class 1 % Image ≤ 0, totalTime > 216, Repeated ≤ 0.625, height $\leq 1 \longrightarrow$ class 1
4	% Image ≤ 0, % Binary Exec ≤ 0.2352, totalTime > 126, height ≤ 1 $\longrightarrow$ class 1 totalPages ≤ 4, totalTime > 171, Repeated $\leq 0.125 \longrightarrow$ class 1 % Image ≤ 0.3296, Repeated ≤ 0.25, width $> 1 \longrightarrow$ class 0 totalPages > 4, % Image > 0, % Zip ≤ 0, width $\leq 4 \longrightarrow$ class 0 totalTime ≤ 126, GET > 0.8571, width $\leq 7 \longrightarrow$ class 0 Binary Exec $> 0.2352 \longrightarrow$ class 0

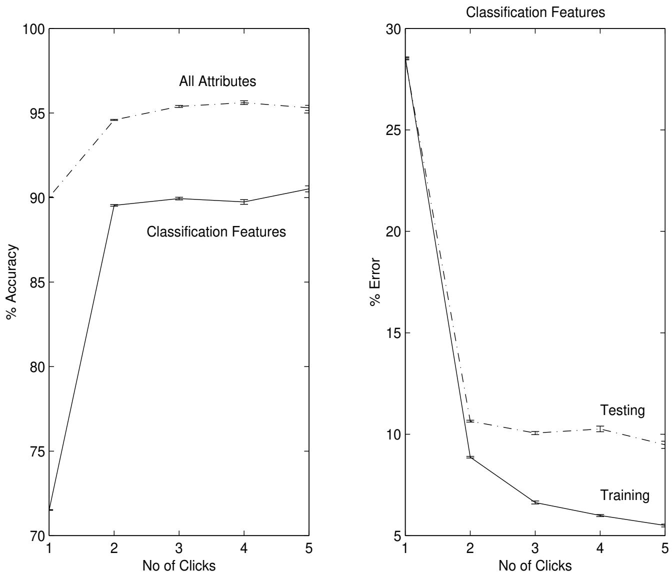

Fig. 3. Accuracies of classification models using different number of clicks. The graph on the left also shows the percent accuracy when all attributes (Table 2) are used. The graph on the right show the average training and testing error rates.

in Table 4 suggests that if the totalTime is less than 10 minutes but there are at least one image file being accessed, then it is most likely a non-robot session. Both the coverage and confidence of this is high.

3. After three clicks, the standard deviation and width attributes play a significant role in discriminating robot sessions. For example, in the absence of image page requests, if the width of the session in the URL space is greater than 4, then it is very likely a robot session. This rule is most likely triggered by Web robots with a breadth-first retrieval strategy.

5 Conclusion

We summarized the above results into the following conclusions :

1. A highly accurate classifier can be built using access features other than obvious attributes such as robots.txt, HEAD request methods and "-" referrer fields. This features can be easily derived from Web server log sessions. Among the most notable features are the % of image files requested, totalTime, width, GET request methods, etc.

The discriminating ability of these features allow us to build predictive models with high accuracy.
2. Based on our proposed set of simple features, our results suggest that Web robots can be detected with reasonably high accuracies after 3 clicks.

Ultimately, our goal is to devise an incremental scheme to infer anonymous Web robots. Our preliminary results are very promising because they indicate that such a scheme can be realized if we have sufficiently large number of training samples with highly accurate class labels. For future work, we plan to incorporate other metrics as defined by W3C Web Characterization Metrics [18] into feature vectors. Our current model can also be improved by incorporating Web content and structure data. For example, we have approximated the width and depth of the session based on their URL address. A better approach would be to compute the width and depth atttributes according to their hyperlink structure. Determination of user clicks can also be improved using Web content and structure information. Another important issue is the different cost of misclassification errors. We hope to address this issue in our future work.

References

1. Altavista search engine. http://www.altavista.com.
2. Email digger. http://www.strayernet.com/webdesign/emailpro.html.
3. evenbetter.com. http://www.evenbetter.com.
4. Extractor pro. http://www.extract.com.
5. Link scan. http://www.elsop.com/linkscan/.
6. Lycos search engine. http://www.lycos.com.
7. Teleport pro. http://www.tenmax.com/teleport/pro/home.htm.
8. Windows 95/98 offline browser tools. http://winfiles.cnet.com/apps/98/offline.html.
9. Xenu's link sleuth. http://home.snafu.de/tilman/xenulink.html.
10. Robot exclusion standard revisited. http://www.kollar.com/robots.html, 1996.
11. M. Balabanovic and Y. Shoham. Learning information retrieval agents: Experiments with automated web browsing. In *On-line Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*, 1995.
12. Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1), 1999.
13. D. Eichmann. Ethical web agents. *Computer Networks and ISDN Systems*, 28(1), 1995.
14. J. Kephart and A. Greenwald. Shopbot economics. In *Agents*, 1999.
15. M. Koster. Guidelines for robot writers. http://info.webcrawler.com/mak/projects/robots/guidelines.html, 1994.
16. M. Koster. A standard for robot exclusion. http://info.webcrawler.com/mak/projects/robots/norobots.html, 1994.

17. M. Koster. Robots in the web: threat or treat. *ConneXions*, 9(4), 1995.
18. B. Lavoie. Web characterization metrics. http://www.oclc.org/oclc/research/projects/webstats/currmetrics.htm, 1999.
19. B. Lavoie and H. F. Nielsen. Web characterization terminology and definitions sheet. http://www.oclc.org/oclc/research/projects/webstats/termsrelease.htm, 1999.
20. H. Lieberman. Letizia: An agent that assists web browsing. In *Proc. of the 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.
21. James Pitkow. In search of reliable usage data on the www. In *Sixth International World Wide Web Conference*, pages 451–463, Santa Clara, CA, 1997.
22. C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
23. Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1–2), 1999.