Modeling of Web Robot Navigational Patterns*

Pang-Ning Tan
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455
ptan@cs.umn.edu

Vipin Kumar
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455
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 may camouflage their entries in the Web server logs. In this paper, we examine the problem of identifying Web robot sessions using standard classification techniques. Due to the temporal nature of the data, the classification model may vary depending on the number of requests made by the Web user or robot. Our goal is to determine the minimum number of requests needed to distinguish between robot and non-robot sessions, with reasonably high accuracy. Our preliminary results show that highly accurate models can be obtained after three requests using a small set of access features computed from the Web server 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. Web robots are often used as resource discovery and retrieval tools for Web search engines [1, 6], shopping comparison robots [3], email collectors [4, 2], offline browsers [8, 7], browsing assistants [11, 19], etc. They are also used by Web administrators for site maintenance purposes (such as mirroring and checking for dead hyperlinks). Some programming languages, such as Python, have provided library functions to facilitate the development of agents with Web crawling capabilities. All of this has led to a situation in which one can no longer ignore 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 input data distribution of 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 algorithm may inadvertantly generate frequent itemsets involving Web pages from different page categories. Such spurious patterns may lead analysts of an e-commerce site to believe that Web users are interested in products from various categories when in fact such patterns are induced by robot sessions. This can be avoided if Web robot sessions are removed from the dataset during preprocessing. Thirdly, the deployment of Web robots usually comes at the expense of other users, as they may consume considerable network bandwidth and server resources.

The main contributions of this paper are as follows :

- We provide a brief survey on the various types of Web robots that are being deployed on the Internet.
- We describe the problem of identifying robot sessions from Web server logs. A typical Web log contains information such as the IP address of the client (Web browser or robot), the date and time a request is made, the request 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 agent information (Table 1). Some robots can be easily identified because they follow the ethical guidelines proposed for robot designers; while others are more difficult to detect.
- We identified several distinguishing features that can

*This work was supported by NSF ACI-9982274 and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute.

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

be used to characterize the access patterns of Web robots. These features can be readily computed from the Web server logs.

- The access features are used to construct classification models that will distinguish between robot and non-robot sessions. However, due to the temporal nature of the data, the classification model may change depending on the number of Web page requests made by the Web user or robot. This is because some of the features may not be computable when the number of requests are small. Hence, it is desirable to know the minimum number of requests needed to identify robot sessions with reasonable accuracy.

2. WEB ROBOTS : OVERVIEW

2.1 Types of Web Robots

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

Eichman [13] divides Web robots into two distinct categories : 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]) and agents designed to accomplish a specific task (such as browsing assistants or hyperlink checkers). Search engine robots often use breadth-first retrieval strategy 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, 19]. These agents often rely on an underlying Web spidering mechanism to pre-fetch new pages, and a learning module to predict the relevance of these pages. Unlike search engine robots, the browsing assistant robots adopt a more direct search strategy focussing only 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 method to request information about a particular Web page. Unlike the GET method, the response message to a HEAD request contains only the Web page meta-information and does not involve a full transfer of the Web document.

Kephart et al.[14] defined 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 available on the Web. These robots are more interested in traversing 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. Whenever a robot visits a Web site, it should request the robots.txt file first. This file contains information about which documents can be accessed by the robot and which are forbidden.

Robots that follow the proposed guidelines can be easily identified because (1) they access the robots.txt file before downloading other documents, and (2) they use the User-agent and From fields in the HTTP request header message to declare their identity to the Web server. For example, the last three log entries in Table 1 correspond to accesses by Web robots. The robot corresponding to rows 4 and 5 can be easily identified via its agent field while the robot corresponding to the last row is identified by the page it has requested, i.e. robots.txt. Unfortunately, not every Web robots obey these noble guidelines. Some robots conceal their identities in various 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 requested by one of these concurrent sessions. As a result, if robot detection is based on access to this file alone, the rest of the concurrent robot sessions may go undetected; (3) by having multiple agent information or using the same agent information as conventional Web browsers (row 6 of Table 1).

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 intentionally want to remain anonymous while traversing a Web site. Therefore, other means of robot identification are needed. In this paper, we will investigate how Web robots can be detected based on information derived from the Web server logs. Our goal is to construct a classification model capable of detecting the presence of Web robot sessions in a fast and accurate manner. Such a model can be used to predict whether the current session belongs to a Web robot or a human user. However, since the amount of information available changes as more pages are being requested by a Web client (browser or robot), the classification model may change depending on the number of requests made by the client. Here, we shall use the term *request* loosely. For Web users, a request refers to all the pages transferred to the browser as a result of a single user click action. For example, the first three entries in Table 1 belongs to the same request. For Web robots, an HTML page along with any other documents embedded within the same page constitute a single request.

Accuracy of the induced model is a crucial requirement. For e-commerce Web sites, the cost of misclassifying user sessions may far outweigh that of robot sessions. Intuitively, the accuracy of the classification model should improve as

Table 1: Example of a Web server log (some fields such as total number of bytes transferred and status of the request have been omitted for brevity).

IP Address	Timestamp	Method	Requested Page	Protocol	Referrer Page	User Agent
171.64.68.115	10/Apr/2000 01:00:10	GET	http://www-users.cs.umn.edu/~kumar/	HTTP/1.0	-	Mozilla/4.61 [en] (WinNT; I)
171.64.68.115	10/Apr/2000 01:00:10	GET	http://www-users.cs.umn.edu/~kumar/krn.gif	HTTP/1.0	http://www-users.cs.umn.edu/~kumar/	Mozilla/4.61 [en] (WinNT; I)
171.64.68.115	10/Apr/2000 01:00:10	GET	http://www-users.cs.umn.edu/~kumar/book.gif	HTTP/1.0	http://www-users.cs.umn.edu/~kumar/	Mozilla4.61 [en] (WinNT; I)
160.94.1.194	10/Apr/2000 01:43:05	GET	http://www.cs.umn.edu/robots.txt	HTTP/1.0	-	Ultraseek
160.94.1.194	10/Apr/2000 01:43:05	GET	http://www.cs.umn.edu//~heimdahl/csci5802	HTTP/1.0	-	Ultraseek
212.27.205.29	10/Apr/2000 04:13:27	GET	http://www.cs.umn.edu/robots.txt	HTTP/1.0	-	Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)

more data is available. Hence, we need to determine the minimum number of requests needed to obtain classification models with acceptable accuracy. This is because early detection will prevent more resources being tied up by uncooperative Web robots. Speed is another important property. The classification model should rely on features that can be derived easily from the Web logs.

3. METHODOLOGY

The robot detection problem can be decomposed into the following subproblems:

1. Preprocess the Web server logs to identify server sessions.
2. Identify and extract the set of features that can be used to characterize the sessions.
3. Labeling of sessions as robot or non-robot sessions.
4. Construct the classification models.
5. Evaluate the classification models.

3.1 Preprocessing

During preprocessing, individual log entries are aggregated into server sessions according to the IP Address and agent information [20, 12]. New sessions are also identified using a 30-minute inter-session timeout period. Within each session, the log entries are grouped into separate requests where each request may correspond to an individual user click or a single robot request. The best way to do the aggregation is by parsing the entire site files for embedded links to be associated with each HTML page. However, this may not be feasible for e-commerce Web sites having dynamic HTML pages. Currently, we approximate the individual requests heuristically based on the timestamp, referrer fields and type of the requested page (e.g. image or audio pages, etc). For example, if two successive log entries of a particular session have the same referrer field (but not equal to "-"), then they are associated to the same request (e.g. rows 2 and 3 of Table 1). Another heuristic we can use is if both log entries involve image pages and have a "-" referrer field (within a duration of 5 seconds), they will most likely belong to the same request. We are still exploring other heuristics for improving the request identification problem.

3.2 Feature Selection and Session Labeling

After sessionization, 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. The computation of temporal attributes such as TOTALTIME and and AVGTIME is illustrated in Fig. 1. Note that the AVGTIME attribute is not computable unless there are more than one request. The STDEVTIME attribute is meaningless unless there are more than two requests. The WIDTH and DEPTH attributes are computed by constructing a representative graph based on the pathname 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) in the graph. Therefore, a session that contains requests for {/A, /A/B, /C, /D} will have a WIDTH = 3 and a DEPTH = 2. The NIGHT attribute was chosen based on the assumption that ethical robots would access a Web site during its low-usage period (such as at night). As a result, the value of this attribute is determined from the server local time rather than the client's local time.

Some of the attributes in Table 2 are used as access features while others are used to assign the appropriate class label to a server session. The assignment of class labels can be done in the following way :

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 server session belongs to a known Web robot, then Class = 1. An initial list of agent information for known Web robots is needed.

Table 2: Summary of attributes derived from server sessions. The attributes are used for class labeling (denoted as Labeling) 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 the robots.txt file is accessed.	Labeling
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	AVGTIME	Average time between clicks (approx.).	Feature
13	STDEVTIME	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 ≥ 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.	Labeling
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 requests).	Feature
24	REFERRER = "-"	% of requests with referrer = "-"	Labeling

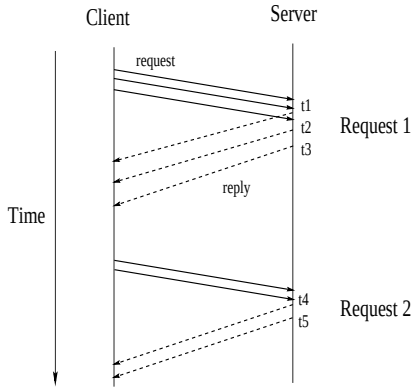

Figure 1: This session contains two requests. $t1$, $t2$, $t3$, $t4$ and $t5$ are the timestamps recorded in the server logs. The value of the temporal attributes for this session are: TOTALTIME $= t5 - t1$, AVGTIME $= t4 - t3$, STDEVTIME $= 0$.

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. 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 (e.g. rows 4 and 5 of Table 1) then there is a strong possibility that the client is a Web robot, as long as the number of requests is large. If number of requests is small, the session can be created by a Web user. For example, if a user supplies the URI of a Web document by typing directly into the address window or clicking on a bookmark, the referrer field of the corresponding Web log entry is "-". Later, we will show that the number of Web robot sessions generated using this heuristic is quite small compare to sessions generated by other means. Thus, the accuracy of the models will not be affected if we ignore the two heuristics.

3.3 Classification

In this paper, we have used the C4.5 algorithm [21] to construct the classification models. C4.5 is a widely used classification algorithm because it can induce models in the form of decision trees and rules that are easier to comprehend compare to other algorithms (such as Naive Bayesian or Support Vector Machines). Since the number of requests vary from one session to another, it is not sufficient to generate a single classification model for all sessions. This is because some of the features are not computable when the number of requests are small. A better way is to induce classification models after each request. Hence, we need to generate different datasets for every number of request. For instance, the dataset for one request is generated from all the server sessions. Sessions with more than one request will be truncated by computing the feature values up to their first request. For dataset with two requests, we ignore all single request sessions, and consider only sessions with at least two requests. Again, sessions with larger number of requests will be truncated so that the feature vector contains only information up to the second request. From these classification models, we can determine the minimum number of requests required to detect Web robot sessions with reasonable accuracy.

3.4 Evaluation

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 :

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

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

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] :

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

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 sessions have more than one request. Therefore, even though our dataset is not labeled based on the HEAD attribute, we believe the the error of mislabeling sessions due to this attribute 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 request and 79 of them have more than 10 requests. Since the REFERRER = "-" heuristic is useful only when the number of requests are large, again the accuracy of our models will not be affected if we ignore this heuristic.

A summary of our datasets 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. On the other hand, 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 then used to induce classification models for each dataset. The sampling and model building procedure is repeated 20 times.

Figure 2 illustrates the correlation [2] between each attribute and the class label for various number of requests. The following results are observed from these graphs:

1. As expected, the class labeling attributes (i.e. HEAD, ROBOTS.TXT, REFERRER) exhibit strong positive correlation with the class labels. Although these attributes can be used to detect ethical robots, they can also be easily manipulated by other robot designers.

2. After two requests, both the TOTALTIME and AVGTIME have rather strong positive correlation with the class label. The % HTML attribute also becomes increasingly important.

3. After three requests, the correlation plot looks quite similar as before, except that the STDEVTIME attribute becomes more important.

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.

Figure 3 illustrates the overall classification accuracies for various models induced from the dataset. Our results show that after two requests, the accuracy improves from 71% to almost 90%. The main reason for such a dramatic improvement is due to the addition of the TOTALTIME and

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

Table 3: Summary of dataset parameters and results of experiment for various number of requests.

Number of Requests	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

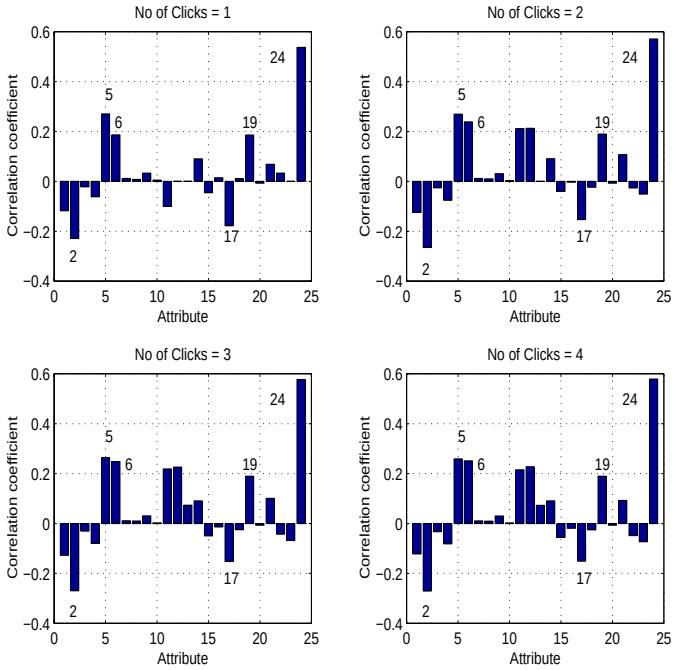

Figure 2: Correlation between access attributes and the Robot class label for various number of requests. The labels accompanying some of the vertical bars indicate their attribute ids (Table 2).

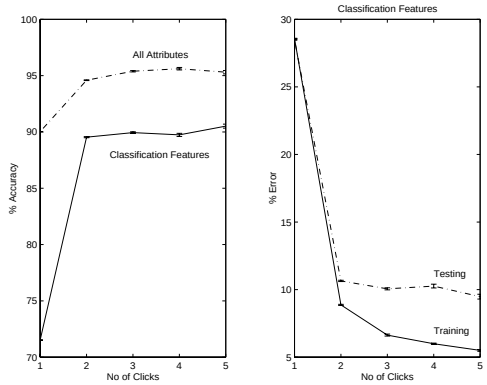

Figure 3: Accuracy of classification models for different number of requests.

AVGTIME attributes. Our precision and recall results consistently reach above 89 % after more than one request. The improved precision at two requests indicates that the addition of AVGTIME and TOTALTIME attributes reduce the amount of false positives due to misclassification of non-robots. The overall accuracy would level off after 3 requests until it begins to deteriorate after 5 requests. This can be explained by the fact that the sample size may not be large enough to induce accurate models. Improvement in the training error for large number of requests can be attributed to overfitting of the dataset; it does not add any predictive power on the test set.

5. CONCLUSION

In summary, our results show that highly accurate models can be built using access features other than obvious attributes such as ROBOTS.TXT, HEAD and REFERRER = "-". These features can be easily derived from the Web server logs. Our results also suggest that Web robot sessions can be detected with reasonably high accuracy after 3 requests.

For future work, we would like to devise an incremental scheme to infer anonymous Web robots. Our preliminary results look 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. We also plan to incorporate other metrics as defined by W3C Web Characterization Metrics [18] into feature vectors. Our current methodology can also be improved by incorporating Web content and structure data. For example, request identification can be improved using both type information. Another important issue is the different cost of misclassification errors. We hope to address this problem in our future work.

6. 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] H. Lieberman. Letizia: An agent that assists web browsing. In *Proc. of the 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.

[20] 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.

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

[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.