Crawler Detection: A Bayesian Approach

Athena Stassopoulou
Department of Computer Science
Intercollege
P.O. Box 24005, 1700 Nicosia, Cyprus
Email: stassopoulou.a@intercollege.ac.cy

Marios D. Dikaiakos
Department of Computer Science
University of Cyprus
P.O. Box 20537, 1678 Nicosia, Cyprus
Email: mdd@cs.ucy.ac.cy

***Abstract*— In this paper, we introduce a probabilistic modeling approach for addressing the problem of Web robot detection from Web-server access logs. More specifically, we construct a Bayesian network that classifies automatically access-log sessions as being crawler- or human-induced, by combining various pieces of evidence proven to characterize crawler and human behavior. Our approach uses machine learning techniques to determine the parameters of the probabilistic model. We apply our method to real Web-server logs and obtain results that demonstrate the robustness and effectiveness of probabilistic reasoning for crawler detection.**

I. Introduction

Crawlers (a.k.a. robots, wanderers, spiders, or harvesters) are programs that traverse the Web autonomously in order to retrieve content and knowledge from the Web on behalf of various Web-based systems and services. The growing need for advanced information- and knowledge-retrieval tools on the Web has led to a remarkable increase in the number of crawlers actively engaged in various types of Web harvesting, and has turned crawlers into an essential component of the Web infrastructure. Currently, there is a growing need to distinguish robots from humans when analyzing the HTTP-request arrivals at Web servers of interest in order to avoid problems such as *click fraud*[1] or the overloading of busy Web servers by agressive robots. To this end, we need to be able to isolate the behavior of robots from that of the general population of (human) Web users. However, the openness, the lack of central control, the sheer size, and the dynamic nature of the Internet, render the identification and registration of active crawlers and operational search engines a very difficult challenge. Public lists of known-robots' domain names and IP addresses do exist [2], [3], but these lists are neither exhaustive nor up-to-date. Furthermore, it is very hard to detect a Web robot *automatically* from the HTTP activity it induces upon an individual Web server. This difficulty is due to the fact that different crawlers exhibit widely differing behaviors in their navigation and HTTP-traffic patterns [4].

In this paper, we introduce a novel approach that addresses successfully the challenging problem of automatic crawler detection using probabilistic modeling [5], [6]. In particular, we construct a *Bayesian network* that classifies automatically access-log sessions as being crawler- or human-induced. To this end, we combine various pieces of evidence, which, according to earlier studies [4], [7], were shown to distinguish the navigation patterns of crawler and human user-agents of the World-Wide Web. Our approach uses machine learning to determine the parameters of our probabilistic model. The resulting classification is based on the maximum posterior probability of each class (crawler or human), given the available evidence.

The remaining of this paper is organized as follows. In Section II we present an overview of our approach and describe its pre-processing steps. The proposed Bayesian network classifier is introduced in Section III. An extensive discussion of our experiments and experimental results is given in Section IV, and we conclude in Section V.

II. Overview

A. Session Identification

The goal of our work is to classify automatically an HTTP user-agent either as a crawler or a human, according to the characteristics of that agent's visit upon a Web server of interest. These characteristics are captured in the Web-server's *access logs*, which record the HTTP interactions that take place between user agents and the server. A typical access-log file is comprised of thousands of entries, sorted by the time the request was posted; each entry represents an HTTP request arriving at the Web server from some user agent along with the server's reply. Each access-log captures a number of sessions, where each *session* is a sequence of requests issued by a single user-agent on a particular server, i.e. the "click-stream" of one user [8]. A session ends when the user completes her navigation of the corresponding site. *Session identification* is the task of dividing an access log into sessions. This is usually performed by grouping all requests that have the same IP address and using a *timeout* method to break the click-stream of a user into separate sessions [8]. According to the *timeout* method, a session is defined as a sequence of requests issued by a single user-agent, such that the time-interval between two consecutive requests is shorter than a pre-defined threshold. A drawback of this method is that it is hard to determine a proper threshold-value, as different user-agents exhibit different navigation behaviors. Usually, a 30-minutes period is adopted as the threshold in Web-mining studies [8].

Nevertheless, in our experiments we noticed that using the 30-minute threshold as the only criterion for breaking the click-stream into sessions was not sufficient. We observed the sessions extracted when using the 30-minute value and noticed that, for longer sessions (in terms of number of requests), click-streams belonging to a semantically continuous navigation activity were split into separate sessions. To

0-7695-2649-7/06/$20.00 (c) IEEE

cope with this issue, we introduce a procedure which adapts the threshold value dynamically, according to the number of session requests. In particular, for sessions with less than r_{max} requests, we set the threshold value to t_1. For sessions with more than r_{max} requests, we increase the threshold value to $t_2 > t_1$. In other words, we allow a bigger time lapse between consecutive requests for larger sessions. By trying various threshold values and studying the resulted sessions, we determined that setting r_{max} to 100, t_1 to 30 minutes and t_2 to 60 minutes gave the best results. Undoubtedly, there is inherent uncertainty in this approach and in any method used to identify Web sessions based on originating IP addresses. For instance, requests posted from the same IP address during the same time period do not come necessarily from the same user-agent [8]: sometimes, different user-agents may use the same IP address to access the Web (for instance, when using the same proxy server); in those cases, their activity is registered as coming from the same IP address, even though it represents different users. Also, session identification based on the heuristic timeout method carries a certain degree of uncertainty regarding the end of a user-agent's navigation inside a Web site of interest. Uncertainty in the data and the actual detection problem itself are the reasons that we believe a probabilistic approach is an ideal application to this problem.

B. System Overview

Our system uses training to learn the parameters of a probabilistic model (Bayesian network) that classifies the user-agent of each Web session as crawler or human. To this end, the system combines evidence extracted from each Web session. Classification is based on the maximum posterior probability given the extracted evidence. The classification process comprises three main phases: (i) Access-log analysis and session identification; (ii) Learning, and (iii) classification. An overview of the functionality of our crawler-detection system is given in Algorithm 1.

1) Access-log analysis and session identification.
2) Session features are selected to be used as variables (nodes) in the Bayesian network. These include:
   a) Maximum sustained click-rate.
   b) Session duration.
   c) Percentage of image requests.
   d) Percentage of pdf/ps requests.
   e) Percentage of requests with 4xx response code.
   f) *robots.txt* file requests.
3) Construction of the Bayesian network structure.
4) Learning:
   a) Labeling of the set of training examples. At this step, sessions are classified as crawler- or human-initiated sessions to form the set of examples of the two classes.
   b) Learning the required Bayesian network parameters using the set of training examples derived from step 4a.
   c) Quantification of the Bayesian network using the learned parameters.
5) Classification: we extract the features of each session and use them as evidence to be inserted into the Bayesian network model. A probability of each session being a crawler is thus derived.

Algorithm 1: Crawler detection system

III. A Bayesian Network Classifier

A. Feature Selection and Labeling Training Data

We base our selection of features on our earlier characterization study of crawler behavior [4], [7]. These features (attributes) are extracted for each session and provide the distinguishable characteristics between Web robots and humans. They are as follows: (i) *Maximum sustained click rate*: This feature corresponds to the maximum number of HTML requests ($clicks$) achieved within a certain time-window inside a session. The intuition behind this is that there is an upper bound on the maximum number of clicks that a human can issue within some specific time frame t, which is dictated by human factors. To capture this feature, we first set the time-frame value of t and then use a *sliding window of time t* over a given session in order to measure the maximum sustained click rate in that session. The *sliding window* approach starts from the first HTML request of a session and keeps a record of the maximum number of clicks within each *window*, sliding the window by one HTML request until we reach the last one of the given session. The maximum of all the maximum clicks per window gives the value of this attribute/feature. (ii) *Duration of session*: This is the number of seconds that have elapsed between the first and the last request. Crawler-induced sessions tend to have a much longer duration than human sessions. Human browsing behavior is more focused and goal-oriented than a Web-robot's. Moreover, there is a certain limit to the amount of time that a human can spend navigating inside a Web site. (iii) *Percentage of image requests*: This feature denotes the percentage of requests to image files (e.g. jpg, gif). Our earlier study showed that crawler requests for image resources are negligible [4]. In contrast, human-induced sessions contain a high percentage of image requests since the majority of these image files are embedded in the Web-pages they are trying to access. (iv) *Percentage of pdf/ps requests*: This denotes the percentage requests seeking postscript(ps) and pdf files. In contrast to image requests, some crawlers, tend to have a higher percentage of pdf/ps requests than humans [4]. (v) *Percentage of 4xx error responses*: Crawlers have a higher proportion of 4xx error codes in their requests. This can be explained by the fact that human users are able to recognize, memorize and avoid erroneous links, unavailable resources and servers [4]. (vi) *Robots.txt file request*: This feature denotes whether a request to the *robots.txt* file was made during a session. From our studies, we also noticed that the majority of crawlers do not request the $robots.txt$ file. Therefore, a strong feature for determining the identity of a session as crawler-induced is the access to the $robots.txt$.

These features form the nodes (variables) of our Bayesian network. The Bayesian network framework enables us to combine all these pieces of evidence and derive a probability for each hypothesis (crawler vs. human) that reflects the total evidence gathered.

Our *training dataset* consists of a number of sessions, each one with its associated label (crawler or human). Since

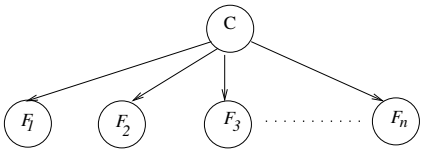

Fig. 1. A Bayesian network used as a classifier

the original dataset contained thousands of sessions, it was prohibitively large to be labeled manually. Therefore, we developed a semi-automatic method for assigning labels to sessions, using heuristics. All sessions are initially assumed to be $human$. Then, we took into account a number of heuristics to label some of the sessions as crawlers: (i) IP addresses of known crawlers; (ii) The presence of HTTP requests for the Robots.txt file; (iii) Session duration values extending over a period of three hours; (iv) An HTML-to-image request ratio of more than 10 HTML files per image file. More information on these heuristics is given in [9]

It should be noted that we only use the first of the heuristics above to determine conclusively the label of the session as $crawler$. The other heuristics are used to give a recommended labeling of the session as $crawler$. These latter sessions are then manually inspected by a human expert to confirm or deny the suggested crawler labeling. By this semi-automatic method we aimed at minimizing the noise introduced in our training set.

B. Network Structure

Bayesian Networks [10], [6], [5] are directed acyclic graphs in which the nodes represent multi-valued variables, comprising a collection of mutually exclusive and exhaustive hypotheses. The arcs signify *direct dependencies* between the linked variables and the direction of the arcs is from *causes* to *effects* [1]. The strengths of these dependencies are quantified by conditional probabilities. More specifically, each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node, where $Parents(X_i)$ denotes the parent variables of X_i. This conditional probability distribution, which defines the *conditional probability table* of the variable, describes the probability distribution of the variable for each configuration of its parents.[2] The graph encodes that each node is conditionally independent of its non-descendants, given its parents [10].

Naive Bayes is a special case of a Bayesian network, where a single cause (the class) directly influences a number of effects (the features) and the cause variable has no parents. This network is shown in figure 1. Again, the independence assumption encoded by this model is that each feature is conditionally independent given the class value.

Considering Figure 1, assume that F_1, F_2,..,F_n are n features and f_i represents the value of feature F_i. Assume also that C is the class variable and let c represent a possible value (label) of C. Using Bayes rule and the conditional independence assumption, we can derive the *posterior probability* of each class label $c \in C$, i.e. the probability of the class label given the features observed, to be given by the formula:

$$P(c|f_1, ..., f_n) = \frac{P(c)\prod_{i=1}^{n} P(f_i|c)}{P(f_1, f_2, \cdots, f_n)} \quad (1)$$

Finally, the class variable C is assigned the label that gives the maximum posterior probability given the features observed. More specifically:

$$class = argmax_{c \in C} P(c)\prod_{i=1}^{n} P(f_i|c) \quad (2)$$

Notice that the denominator in equation 1 is a constant and can be ignored in the last step.

The proposed Bayesian Network for crawler detection has the structure shown in Figure 1. Before we explain the reasoning behind this structure, we first give an interpretation of each of the nodes. Each child node corresponds to one of the features we presented earlier in section III-A. The root node represents the $class$ variable.

All nodes used in the network have been abbreviated as follows:

- *Class*: The classification of the session. This variable takes two values: $robot$ or $human$. This is the root node of the network of Figure 1.
- *Clicks*: Maximum number of clicks within a certain time frame. This variable takes values in the range $[0, .., maxClicks]$ where $maxClicks$ is determined by our training data.
- *Duration*: The number of seconds between the first and the last request. It takes values in the range $[0, .., maxDuration]$ where $maxDuration$ is determined by our training data.
- *Images*: Percentage of requests to image files (e.g. jpg, gif). This takes values in the range $[0, .., 100]$.
- *PDF/PS*: Percentage requests to postscript(ps) and pdf files. This takes values in the range $[0, .., 100]$.
- *Code 4xx*: Percentage of 4xx error responses. This takes values in the range $[0, .., 100]$.
- *Robots.txt*: This a variable has only two values: 1 or 0. The value 1 means that the session included a request to the $robot.txt$ file, otherwise the value is 0.

The network structure indicates that the class in which the session belongs (i.e. crawler or human), "causes" its features (attributes) and thus the direction of the arrow from class to feature. This model encodes that the "effect" variables are *conditionally independent* given the cause. For more details on causality and learning causal structures see [6], [5].

Regarding the "weight" that each piece of evidence bears on the classification (i.e. the fact that a certain feature may be more significant than an other in determining the classification of a session), is implicitly encoded in the conditional

[1] If there is an arc from node X to node Y, then X influences (or causes) Y. In such a case X is the *parent* of Y

[2] In this paper we consider Bayesian networks with discrete variables.

probability distributions that relate a child (the feature in this case) with the parent (the class).

Having defined the structure of the network, we now have to (i) Discretize all continuous variables; (ii) Define the conditional probability tables that quantify the arcs of the network. In the next two section we show how we use machine learning to achieve the above tasks.

C. Learning Network Parameters

The learning phase of the system uses the training data that have been created as described in section III-A. The training data set consists of a number of sessions, each one with its associated label (crawler or human). For each of these sessions, we obtain the values of each of the features, described in section III-B above, and which are represented as nodes in the Bayesian network. We use the data for variable quantization, based on the entropy, as well as for learning the conditional probability tables, as described in the next two sections.

1) Variable Quantization: Since, in this implementation, the Bayesian Network is developed for discrete variables, the continuous variables need to be quantized- divided into meaningful states (meaningful in terms of our goal, i.e. to detect crawlers). One well-known measure which characterizes the purity of the class membership of different variable states is *information content* or *entropy* [11]. The procedure used here was as follows: We observe the values of the variables for a set of $Crawler$ and $Human\ session$ examples. For a given quantization into c classes, the entropy is given by:

$$E = \sum_{i=1}^{c} -P_{Ci}log_2P_{Ci} - P_{Hi}log_2P_{Hi} \quad (3)$$

where P_{Ci} is the probability of crawler sessions in class i and P_{Hi} is the probability of human sessions in class i. The entropy is then weighted by the fraction of examples that belong in each interval. The number and range of classes which result in the minimum total weighted entropy are chosen to quantize the variable.

This minimum entropy principle was applied on all the continuous variables (nodes), i.e. on five out of the six features presented in section III-B: $Clicks$, $Duration$, $Images$, PDF/PS and $Code\ 4xx$.

2) Conditional Probabilities: Having constructed the network nodes, we need to define the conditional probabilities which quantify the arcs of the network. More specifically, we need to define the $a\ priori$ probability for the root node, $P(Class)$ as well as the conditional probability distributions for all non-root nodes: $P(Clicks|Class)$, $P(Duration|Class)$, $P(Images|Class)$, $P(PDF/PS|Class)$, $P(Code\ 4xx|Class)$, with variables abbreviated as in section III-B. Each of these tables gives the conditional probability of a child node to be in each of its states, given all possible parent state combinations.

We derived these probabilities from statistical data. For example, the conditional probability of $Duration$ being in class (state) 1 given $Class = Crawler$, is determined from data, by counting the number of Crawler examples with a duration within class 1, and so on.

D. Classification

Once the network structure is defined and the network is quantified with the learned conditional probability tables, we proceed with the classification phase of our crawler detection system.

For each session to be classified, we extract the set of six features that characterize the behavior of clients and that form the variables of our Bayesian Network. An example feature vector, based on the feature description given in III-A, could be (17, 135.5, 67, 2, 0, 0) which can be described as follows: the session in question had reached a pick of 17 clicks (in a pre-set 12-second window), had a session length of 135.5 seconds, 67% of of its requests were to image files, 2% of its requests were to pdf/ps files, there were 0% requests with response code greater than 400 and, finally, that the robots.txt file was not requested (indicated by the last binary value being set to 0). As described above, the network contains only discrete variables whereas the first five of the six features are continuous-valued. Each of these feature values is therefore mapped on to a discrete state according to the ranges derived by the quantization step of section III-C.1.

Following this step, each session is now characterized by six features represented as values of discrete variables corresponding to the Bayesian network. In order to classify a session, each variable in the network is instantiated by the corresponding feature value. The Bayesian network then performs inference and derives the $belief$ in the $Class$ variable, i.e. the posterior probability of the $Class$ to take on each of its values given the evidence (features) observed. In other words we derive: $P(Class = crawler|evidence)$ and $P(Class = human|evidence)$. The maximum of the two probabilities is the final classification given to the session.

IV. Experimental Results

In this section we present the experiments performed in order to apply our methodology and evaluate the performance of our crawler detection system.

A. Training Data sets

For the purposes of evaluating the performance of our crawler detection system, we obtained access logs from two servers of two academic institutions: the University of Toronto and the University of Cyprus (a detailed description of these log files can be found in [4]). The access logs were processed by our log analyzer to extract the sessions. These sessions, the majority being from the University of Toronto, were used for training. Sessions were then labeled using our approach described in section III-A.

The learning stage proved to be challenging task. The problem encountered with this stage is one of $class\ imbalance$ [12], [13], [14]. The data sets present a class imbalance when there are many more examples of one class than of the other. It is usually the case that this latter class,

i.e. the unusual class, is the one that people are interested in detecting. Because the unusual class is rare among the general population, the class distributions are very skewed [12]. In our earlier study [4] we have concluded that crawler activity in access logs amount to less than 10 per cent of the total number of requests.

The problem that arises from training with imbalanced data set is that classifiers tend to be biased toward the majority class, i.e. the class with the largest number of examples. In the case of the Naive Bayes classifiers, the prior probability in the majority class overshadows the differences that exist in the conditional probability entries that quantify the relationship between feature and class variables.

To tackle the problem of imbalanced data sets we use $resampling$. More specifically we adopted two resampling approaches in our experiments: random oversampling and random undersampling. In the former method, the minority cases, i.e. crawler sessions, are randomly chosen for duplication until the ratio of majority to minority reaches a desirable level. In the latter method of random undersampling, the majority cases, i.e. human sessions, are randomly eliminated until the ratio is at the desirable level. We performed 5 experiments, based on resampling (both oversampling and undersampling) at various ratios.

Table I shows the number of Crawler and Human sessions in each of the 5 training data sets created via resampling. The last column shows the prior probability distributions of variable $Class$, considering the distribution of sessions actually used for training.

We constructed five Bayesian network classifiers, one for each experiment. The networks had the same structure but differed in their parameters, i.e. prior probabilities, conditional probability tables and quantization ranges. Each time a new training data set was introduced, new network parameters were derived using training on the new set.

B. Testing the system

A different access log, from the ones not used during training, was randomly chosen for testing. Since the majority of the sessions used for training were extracted from the University of Toronto log, we have chosen a different institution server altogether to evaluate our detection system. This access log used for testing was obtained from the University of Cyprus and spanned a period of one month. A human expert did an entirely manual classification of each session, extracted by our log analyzer from this the testing set, in order to provide us with the ground truth by which we were to evaluate our classifier's performance. It should be noted that we did not do any resampling for the testing.

We tested the performance of all five Bayesian networks (one for each data set), on the same testing dataset[3]. The testing set contained 685 actual human sessions and 99 actual crawler sessions, as labeled by an independent human expert. Throughout this section we will refer to the 5 classifiers as follows: (i) Classifier $C1$: Obtained using learning of Data set 1 (no resampling); (ii) Classifier $C2$: Obtained using learning of Data set 2 (oversampling to 15%); (iii) Classifier $C3$: Obtained using learning of Data set 3 (oversampling to 50%-equally represented classes); (iv) Classifier $C4$: Obtained using learning of Data set 4 (undersampling to 85%); (v) Classifier $C5$: Obtained using learning of Data set 5 (undersampling to 50%-equally represented classes).

[3]The networks were implemented using the $Ergo^{TM}$ tool [15]

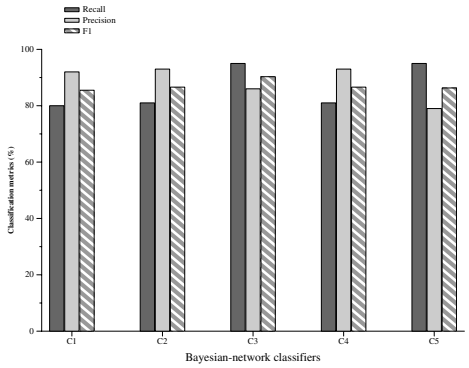

Fig. 2. An evaluation of the crawler-detection classifiers.

Two metrics that are commonly applied to imbalanced datasets to evaluate the performance of classifiers is $recall$ and $precision$. These two metrics are summarized into a third metric known as the F_1-measure [16]. These metrics, applied to our problem, are defined as follows:

$$Recall(R) = \frac{\text{No. of Crawler sessions correctly classified}}{\text{No. of actual crawler sessions}}$$

$$Precision(P) = \frac{\text{No. of Crawler sessions correctly classified}}{\text{No. of predicted Crawler sessions}}$$

The F_1-measure is the harmonic mean between recall and precision. It summarizes the two metrics into a single value, in a way that both metrics are given equal importance. Recall and precision should therefore be close to each other, otherwise the F_1-measure yields a value closer to the smaller of the two.

The values of recall, precision and F_1-measure obtained by classifiers $C1, \ldots, C5$ are given in Table II and plotted in Figure 2.

Classifier	Recall	Precision	$F_1 - measure$
C1	0.80	0.92	0.855
C2	0.81	0.93	0.866
C3	0.95	0.86	0.903
C4	0.81	0.93	0.866
C5	0.95	0.79	0.863

TABLE II
EVALUATION METRICS OF EACH BAYESIAN NETWORK CLASSIFIER.

As it can be seen from table II, our crawler detection system yields promising results with both recall and precision being above 79% in all experiments performed. The lowest $F1$-measure is obtained by C1 when we train the system with

Data Set No.	No. Distinct Humans	No. Distinct Crawlers	No. Humans used in training	No. Crawlers used in training	Prior Probabilities: (Human, Crawler)
1	10106	988	10106	988	(0.91, 0.09)
2	10106	988	10106	1784	(0.85, 0.15)
3	10106	988	10106	10106	(0.5, 0.5)
4	10106	988	5599	988	(0.85, 0.15)
5	10106	988	988	988	(0.5, 0.5)

TABLE I

DATA SETS USED FOR FIVE EXPERIMENTS WITH AND WITHOUT RESAMPLING

the dataset without resampling. The prior probability of a session to be *Human* in that dataset was 91% and the classifier was therefore biased towards humans. It missed only 7 out of the 685 *Human* sessions but sacrificed recall, by missing 20 out of the 99 actual *Crawler* sessions. By resampling so that the *Crawler* class amounts to 85% of the sessions (either via oversampling as in C2 or by undesampling as in C4) we have slightly improved results compared to C1. Both C2 and C4 have the same precision and recall. The best results are obtained by C3, which was trained using oversampling of *Crawlers* so that they reach the number of *Human* examples in the original set. The recall, i.e. the percentage of crawlers correctly classified increases dramatically to 95%, with 94 sessions correctly classified as *Crawlers* out of 99 actual crawlers. This causes a decrease in precision, which is nevertheless not so dramatic. The same recall as C3 is achieved by C5 which was trained by undersampling *Humans* so that both classes are again, equally represented. However, this caused a significant decrease in precision to 79%, i.e. we have an increase in the number of false positives, i.e. *Humans* incorrectly classified as *Crawlers*. The significant decrease in precision of C5, is not surprising since, with random undersampling there is no control over which examples are eliminated from the original set. Therefore significant information about the decision boundary between the two classes may be lost. The risk with random oversampling is to do over-fitting due to placing exact duplicates of minority examples from the original set and thus making the classifier biased by "remembering" examples that were seen many times. The are other alternatives to random resampling which may reduce the risks outlined above. An investigation and a comparison of the various resampling techniques is beyond the scope of the current paper.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the use a Bayesian network, for detecting Web crawlers from access logs. This Bayesian approach is well suited for the particular domain due to the high degree of uncertainty inherent in the problem. Our system uses machine learning to determine the parameters of the Bayesian network that classifies the user-agent of each Web session as crawler or human. The system combines evidence extracted from each Web session to determine the class it belongs to. The Bayesian network does not merely output a classification label, but a probability distribution over all classes by combining prior knowledge with observed data. We have used resampling to counter the class imbalance problem and developed five classifiers by training on five different datasets.

The high accuracy with which our system detects crawler sessions, proves the effectiveness of our proposed methodology. These results provide a promising direction for future work. We are currently investigating the introduction of additional heuristics for session identification, which is an important pre-processing step of our proposed system. We also plan to introduce other interesting features of Web sessions, such as the navigational semantics of user-agent requests.

REFERENCES

[1] D. A. Vise, "Clicking to Steal. When Advertisers Pay by the Lok, Fraud Artists See Their Chance," Washington Post, April 17, 2005, http://washingtonpost.com.

[2] "Search Engine Watch: Tips about Internet Search Engines and Search Engine Submission," http://searchenginewatch.com/ (last accessed Oct. 2005).

[3] "The Web Robots Pages," http://www.robotstxt.org/ (last accessed Oct. 2005).

[4] M. D. Dikaiakos, A. Stassopoulou, and L. Papageorgiou, "An Investigation of WWW Crawler behavior: Characterization and Metrics," *Computer Communications*, vol. 28, no. 8, pp. 880–897, May 2005.

[5] R. E. Neapolitan, *Learning Bayesian Networks*. Pearson Prentice Hall, 2004.

[6] J. Pearl, *Causality: Models, reasoning and Inference*. Cambridge University Press, 2000.

[7] M. D. Dikaiakos, A. Stassopoulou, and L. Papageorgiou, "Characterizing Crawler Behavior from Web Server Access Logs," in *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003)*, ser. Lecture Notes in Computer Science, A. M. T. K. Bauknecht and G. Quirchmayr, Eds. Springer, September 2003, vol. 2738, pp. 369–378.

[8] 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, January 2000.

[9] A. Stassopoulou and M. D. Dikaiakos, "Human or Crawler? A Probabilistic Reasoning Approach," Department of Computer Science, University of Cyprus, Tech. Rep. TR-06-1, February 2006.

[10] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

[11] T. M. Mitchell, *Machine Learning*. McGraw Hill Companies Inc., 1997.

[12] F. J. Provost and T. Fawcett, "Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions." in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997, pp. 43–48.

[13] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Computational Intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[14] G. M. Weiss, "Mining with rarity: a unifying framework." *SIGKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[15] "Noetic Systems Incorporated," http://www.noeticsystems.com/ergo/index.shtml.

[16] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison-Wesley, 2005.