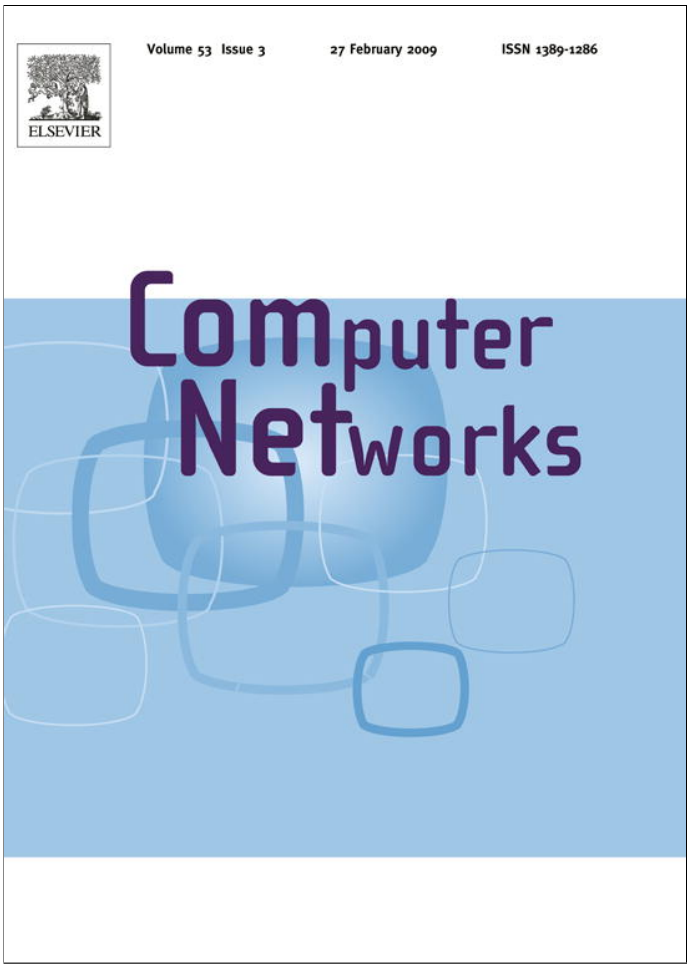
Volume 53 Issue 3
27 February 2009
ISSN 1389-1286
ELSEVIER
Computer Networks

Computer Networks 53 (2009) 265–278

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Web robot detection: A probabilistic reasoning approach

Athena Stassopoulou [a,*], Marios D. Dikaiakos [b]

[a] Department of Computer Science, University of Nicosia, P.O. Box 24005, 1700 Nicosia, Cyprus
[b] Department of Computer Science, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus

ARTICLE INFO

Article history:
Received 15 February 2008
Received in revised form 18 July 2008
Accepted 23 September 2008
Available online 21 October 2008

Responsible Editor: C.B. Westphall

Keywords:
Web crawler detection
Probabilistic reasoning
Bayesian classifiers

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 an adaptive-threshold technique to extract Web sessions from access logs. Then, we apply machine learning techniques to determine the parameters of the probabilistic model. The resulting classification is based on the maximum posterior probability of all classes given the available evidence. We apply our method to real Web-server logs and obtain results that demonstrate the robustness and effectiveness of probabilistic reasoning for crawler detection.

1. Introduction

Crawlers are programs that traverse the Web autonomously, starting from a "seed" list of Web-pages and recursively visiting documents accessible from that list. Crawlers are also referred to as robots, wanderers, spiders, or harvesters; their primary purpose is to discover and retrieve content and knowledge from the Web on behalf of various Web-based systems and services. For instance: *search-engine crawlers* seek to harvest as much Web content as possible on a regular basis, in order to build and maintain large search indexes [11,8]; *shopping bots* crawl the Web to compare prices and products sold by different e-commerce sites; *focused crawlers* seek and acquire Web-pages belonging to pre-specified thematic areas [12]; *profile-driven crawlers* support personalized and added-value services on the Web [16]; *email harvesters* collect email addresses on behalf of email marketing companies or spammers, and *site-specific crawlers* perform various Web-site maintenance chores, such as mirroring Web-sites or discovering their broken links.

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. Consequently, there is a growing need to distinguish robots from humans when analyzing the HTTP request arrivals at Web-servers of interest, for a variety of reasons:

- Typically, the analysis of HTTP-interactions of a Web-server provides a wealth of information about the functionality, the usability, the design, and the popularity of its hosted sites and content [23,35]. The basic premise here is that HTTP-interaction patterns reflect the choices that end-users (customers) make when navigating inside a Web-site. Furthermore, reliable quantitative information that captures end-user navigation choices is the basis for *pay-per-click* advertising, one of the most prevalent and successful Internet-business models [1]. According to press and business reports, the pay-per-click advertising generates 50–98% of Google's revenues [13,40], and is adopted by other search-engine giants like Yahoo! and AskJeeves and by popular blogging services [24]. There is growing concern, however, that this business model can be seriously harmed by *click*

* Corresponding author. Tel.: +357 22841648; fax: +357 22357481.
E-mail addresses: stassopoulou.a@unic.ac.cy (A. Stassopoulou), mdd@cs.ucy.ac.cy (M.D. Dikaiakos).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.09.021

fraud[13,40,5], which involves, among other things, the unwilling or malicious repetitive retrieval of advertisement links by Web robots that do not emanate from well-known or stable IP addresses and Internet domains. Click fraud results in higher fees paid by the advertisers for what are essentially worthless HTTP requests, and has triggered concerns on the viability of the search-engine business model, leading Google's CFO to state that "click fraud is the 'biggest threat' to the Internet economy!" [40].
- The automatic identification of malicious Web robots according to their behavior, rather than their (possibly transient) IP addresses, can also be useful when trying to cope with the *referer spam* problem, which affects search-engine ranking results [6].
- There is also concern that crawler-induced traffic represents a sizeable part of the total HTTP traffic and that crawler activity may result in a performance degradation of busy Web-servers and networking infrastructures, and in increased miss ratios in Web caches. These concerns are supported by the few published papers that investigated Web robot behavior, analyzing the impact of known crawlers upon different Web-servers [7,14,15,28,42].
- Finally, there is a need to distinguish between crawler and human traffic in cases where it is important to protect information of a temporary or sensitive nature, published on intranet Webs, from crawlers that discover inadvertently such information and publish it through search-engine databases.

To address the concerns mentioned above, we need to be able to isolate the behavior of robots from that of the general population of (human) Web users. Distinguishing Web robots from humans will help marketing companies derive more accurate statistics about the impact of online advertising and the interaction that real customers have with e-business sites. Also, it will help Web administrators in estimating the real side-effects of crawler activity on Web-server performance. Finally, it can provide a basis for developing intelligent admission control systems that will protect Web-sites from aggressive or unwanted crawlers. However, the openness, the lack of central control, the sheer size, and the dynamic nature of the Internet, render the identification of active crawlers and operational search engines a very difficult challenge.

In this article, we introduce a probabilistic modeling approach for addressing the problem of automatic Web robot detection from Web-server access logs. 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. The remainder of this paper is organized as follows: the background and related work is given in Section 2. In Section 3 we present an overview of our approach and describe its pre-processing steps. The proposed Bayesian network classifier is introduced in Section 4. An extensive discussion of our experiments and experimental results is given in Section 5, and we conclude in Section 6.

2. Background and related work

2.1. Robot-identification and characterization

Several efforts try to monitor and catalog the increasing number of robots that crawl the Web in order to publicize information about active crawlers, such as their domain names and/or IP addresses [2–4]. Nevertheless, public crawler-lists are neither exhaustive nor up-to-date, since numerous robots initiate their operation or change their identity without explicitly notifying any third-party registries.

Therefore, a few studies that seek to characterize crawler behavior rely on the access logs of individual Web-servers and focus on robots that either emanate from known IP addresses belonging to the domains of established search engines or declare their identity via special HTTP-protocol headers (like `From` or `User-agent`). For example, in [14,15], we analyzed access logs capturing the HTTP traffic of 42–176 days of five different academic sites of three countries (Cyprus, Greece, and Canada) and showed that the accumulated activity of crawlers belonging to five known search engines (google, altavista, inktomi, fastsearch, and citeseer) contributed to approximately 10% of the total HTTP requests arriving at these sites. We also showed that the visit patterns of crawlers have significant differences from those of humans. Similarly, in [42], Ye, Lu, and Li investigated the hourly distribution of request arrivals from five crawlers (google, inktomi, baidu, webfountain and altavista), using a 180-day-long access log from the Web servers of the China Education and Research Network (CERNET), which captures over 1 billion HTTP requests. Their study showed that crawler traffic contributed to only 1.6% of the overall HTTP traffic, which totaled 5,367.4 GB. They noticed, however, that crawlers did not pay attention to server workload and that crawler visits during peak hours of server activity could affect both server performance and crawler efficiency.

The studies presented above do not capture the overall impact of robots, as they analyze the behavior of a handful of publicly-known crawlers, which represent only a subset of the activity of all the active crawlers. Nevertheless, it is very hard to identify Web robots *automatically* from the HTTP activity they induce upon individual Web servers. This difficulty is due to the fact that different crawlers may exhibit widely differing behaviors in their navigation and HTTP traffic patterns. For instance, some robots comply with simple rules of "polite behavior" while visiting a Web-server (e.g., following the *Robots Exclusion Protocol* or avoiding to swamp it with too many HTTP requests), whereas other robots exhibit long click-streams upon the same site and in a very short time-range; many crawlers retrieve textual and HTML content only, whereas others show preference to postscript and pdf formats, and so on.

An effort to identify robots automatically through heuristics was presented by Menascé, Almeida et al. in [28,7]. In these papers, the authors derived a set of heuristic criteria from observations of human and crawler Web navigational patterns. Subsequently, they used these criteria as rules for isolating robot-induced traffic inside the access logs of an online bookstore site. The authors did not provide any assessment on the success and the accuracy of their proposed robot-identification heuristics. Their main goal was to estimate the impact of robot-induced HTTP traffic on the performance of Web infrastructures; their conclusions showed that the identified robots consumed a significant amount of Web-server resources.

In our characterization study [15], we also tried to identify common traits that distinguish robot behavior from the characteristics of the general HTTP traffic. Nevertheless, from that study it became clear that these characteristics exhibit a wide variability for different crawlers and different sites. Therefore, we concluded that it is hard to derive simple, deterministic, and generic heuristics for the detection of robots from Web-server access logs: a rule-based system for classifying sessions would imply a long list of static rules, that are difficult to define and maintain, even by experts. Instead, a system that can *learn* from examples to distinguish between crawlers and humans would provide many benefits over a rule-based approach. Moreover, a system that is able to cope with uncertainty, such as the one presented in this study, provides even further advantages since it can model uncertainty both in the rules and the data, in the form of conditional probabilities. The outcome of such a system that performs inference under uncertainty, is a probability distribution over all possible classes, that indicates how reliable the classification is given the data.

2.2. Our contribution

In this paper, we introduce a novel approach that addresses successfully the challenging problem of automatic crawler detection using probabilistic reasoning [31]. 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 our earlier studies [15,14], 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.

To the best of our knowledge, this is one of the few published studies that propose a crawler detection system, and the only one that uses a probabilistic approach. An alternative approach that is based on decision trees, was proposed by Tan and Kumar in [38]. The authors applied their method with success on an academic access log collected over a period of one month in year 2001. They base their detection on the navigational pattern of the users and use a number of features to built a decision tree using the C4.5 algorithm.

As it will be evident from the following sections, the application of a probabilistic approach such as Bayesian Networks, is well suited for the particular domain, due to the high degree of uncertainty inherent in the problem. The Bayesian Network does not merely output a classification label, but a probability distribution over all classes by combining prior knowledge with observed data. This probability distribution allows decisions to be made about the final classification based on how "confident" the classification is, as demonstrated by the probability distribution. For example, one need not accept weak classifications where the resulting posterior probability is less than a pre-defined minimum.

Bayesian networks have been shown to provide highly accurate classification results in the presence of incomplete or inexact information from multiple sources in a variety of application areas [36,37,33,27,20,10,22].

3. Overview

Our goal is to classify automatically an HTTP user-agent either as a crawler or a human, based on characteristics of that agent's visit upon a Web-server of interest. Information related to the HTTP traffic arriving at a Web-server is usually kept in the Web-server's *access log*. Typical access logs comprise thousands of entries, with each entry representing an HTTP request that arrives at the Web-server from some user-agent, and the Web-server's reply. Access logs are encoded according to the *NCSA Common Log File Format* or the *NCSA Extended Common Log File Format* [25] (for a description see Table 1). Access log entries are

Table 1
NCSA common log file format fields.

Common Log File Format: Remote hostname (IP address or hostname of the client) Remote login name of the user (rfc931) Username as with the user has authenticated himself Date (dd/mmm/yyyy:HH:mm:ss <difference_from_Greenwich>) Request line of the HTTP message HTTP response code returned to the client Bytes returned (body only)
Example (CLF Format): 194.42.7.30 - - [23/Jan/2002:21:21:33 +0200]"GET/images/main/ucy.jpg 0.5in HTTP/1.0" 200 83212
Example (ECLF Format): 194.42.7.30 - - [23/Jan/2002:21:21:33 +0200]"GET/images/main/ucy.jpg 0.5in HTTP/1.0" 200 83212 http://www.ucy.ac.cy/"Mozilla/2.0GoldB1 (Win95; I)"

kept sorted according to the time that each request was posted.

Because of the stateless nature of HTTP, incoming requests are considered and logged as independent events. Therefore, access logs do not contain any information that could relate together requests issued during a single "visit" of one user-agent to the Web-pages of a Web server. Hence, in order to extract the characteristics of a user-agent visit at a Web-server of interest, we need first to process its access logs and infer that agent's Web session. A *Web session* is defined as the sequence of HTTP requests that originate from the same user-agent and arrive at a Web-server within a common time-frame of limited duration. The session starts with the first HTTP request issued by the user-agent and finishes when the user completes the navigation of the corresponding site. In essence, a Web session represents the "click-stream" of a user navigating inside the pages of a particular Web-server [35].

3.1. System overview

The goal of our system is to classify the user-agent of each Web session as crawler or human. The classification process comprises three main phases: (i) Access log analysis and session identification; (ii) Learning, and (iii) classification.

Our system uses training to learn the parameters of a probabilistic model (Bayesian network) that performs the actual classification. For the learning part, the system combines evidence extracted from each Web session. An overview of the training workflow of our crawler detection system is given in the top diagram of Fig. 1. For the training of our classifier, we use Web sessions extracted from a selected access log. Subsequently, we perform a semi-automatic labeling of these sessions as crawler- or human-induced. Then, we compute the values of certain features from the labeled sessions in order to derive our training data. The training data are used to learn the required Bayesian network parameters and to quantify the Bayesian network using the learned parameters. Classification is based on the maximum posterior probability given the extracted evidence. The classification stage extracts the features of each Web session and uses them as evidence to be inserted into the Bayesian network classifier. The classifier outputs the probability of each session being a crawler. The classification stage is summarized in the bottom diagram of Fig. 1.

In the following section we present the heuristic that we use to identify sessions inside Web-server access logs, whereas the other two phases are examined in detail in Section 4.

3.2. Session identification with adaptive thresholds

Session identification is the task of dividing an access log into sessions. Typically, session identification is performed by first grouping all HTTP requests that originate from the same IP address and user-agent, and then using a *timeout* approach to break this grouping into different sub-groups [35], such that the time-lapse between two consecutive sub-groups is longer 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-min period is adopted as the threshold in Web-mining studies [35].

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 cope with this issue, we introduce a procedure which adapts the threshold value dynamically, according to the

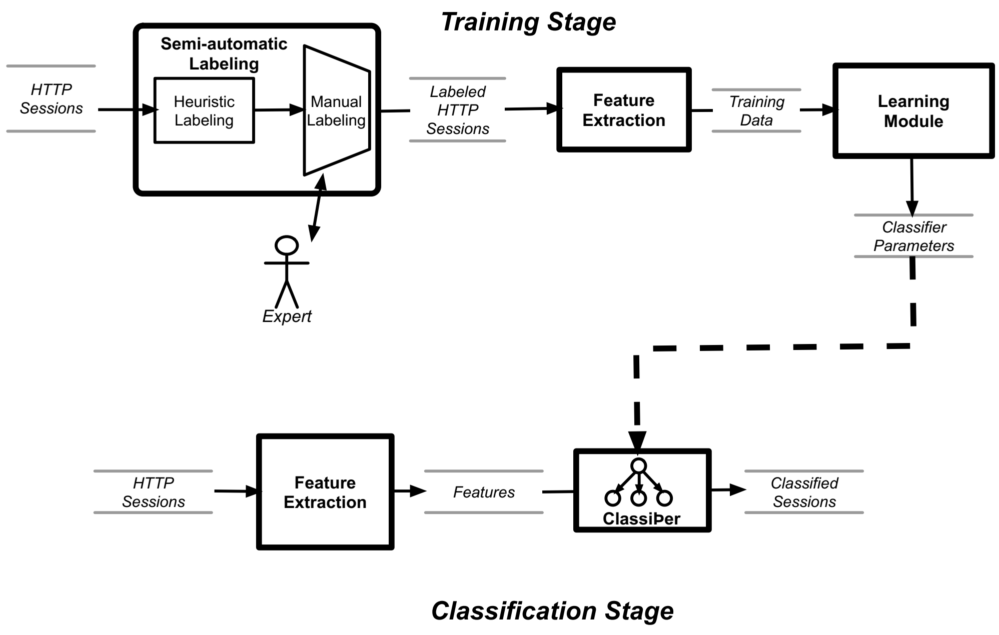

Fig. 1. Overview of the training and classification stages.

number of session requests so far. In particular, for sessions with less than r_{max} requests so far, we set the threshold value to t_1. When the number of requests reaches r_{max}, 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 min and t_2 to 60 min 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 [35]: 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.

4. A Bayesian network classifier

4.1. Feature selection

We base our selection of features on our earlier characterization study of crawler behavior [15,14]. These features (attributes) are extracted for each session and provide the distinguishable characteristics between Web robots and humans. They are as follows:

- *Maximum sustained click rate*: A *click* is a request for an HTML file. This feature corresponds to the maximum number of HTML requests 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. For example, if we set *t* to 12 s and find that the maximum number of clicks within some 12-s time-window inside that session is 36, we conclude that the maximum sustained click rate is 3 clicks per second. This indicates a robot-like rather than a human-like behavior. 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 clicks per window gives the value of this attribute/feature.
- *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.
- *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 [15]. 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.
- *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 [15].
- *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 [15].
- *Robots.txt file request*: This feature denotes whether a request to the *robots.txt* file was made during a session. Web administrators, through the Robots Exclusion Protocol, use a special-format file called *robots.txt* to indicate to visiting robots which parts of their sites should not be visited by the robot. For example, when a robot visits a Web-site, say http://www.foo.com, it should first check for http://www.foo.com/robots.txt. It is unlikely, that any human would check for this file, since there is no link from the Web-site to this file, nor are (most) users aware of its existence. From our studies, we also noticed that the majority of crawlers do not request the `robots.txt` file and so it is the *presence* of a `robots.txt` request in a session that will have the greater impact on it being classified as *crawler*. 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.

4.2. Labeling training data

In this section we present how the training data sample was created. The *training dataset* consists of a number of sessions, each one with its associated label (crawler or human). Our dataset contains thousands of sessions and is therefore 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 use the following heuristics to label some of the sessions as crawlers:

(1) *IP addresses of known crawlers*: We used the IP addresses of HTTP requests recorded in our logs to perform reverse DNS lookups and convert IP

addresses to hostnames. With this mapping we assign IP addresses to crawlers with a well-known set of hostnames. Our log analyzer then prepares a table with all the IP addresses that belong to known crawlers. Each IP address of a training session is compared against this table of IP addresses of known crawlers. If the IP matches, we label the session as a Web robot.

(2) *Robots.txt file*: As explained earlier, any session that includes an HTTP request for the *robots.txt* file is most likely a robot; consequently, we label such sessions as robot-induced.

(3) *Session duration*: If the duration of the session, i.e. the time between the first and the last request, is more than a threshold number of hours, then we assume that this is a crawler-induced session. From our experiments, we found that 3 hours is a good choice of a threshold.

(4) *HTML-to-image request ratio*: As already mentioned before, the results of our earlier study [15,14], showed that humans tend to have a higher number of image requests than crawlers, due to embedded images in HTML files. On the other hand, crawlers have a high number of HTML requests with negligible image requests, since these are usually omitted from downloading by the web robot. This heuristic aims to capture this knowledge about the behavior of crawlers by labeling as crawler, any session with more than 10 HTML files per image file. In other words, if the ratio HTML-to-image in a session is larger than 10, it is more likely that this is a crawler session, and we thus label it as such.

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.

Since we are investigating the *behavior* as evident from the click-stream of a user-agent, it is fair to assume that any session with less than 5 requests in total, is too short to enable labeling. Even by manual inspection, a session with such a few number of requests is almost impossible to classify. We are therefore ignoring sessions that are too small (i.e. with less than 5 requests) from both the labeling and the testing stage.

4.3. Network structure

Bayesian Networks [32,31,30] 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 [31].

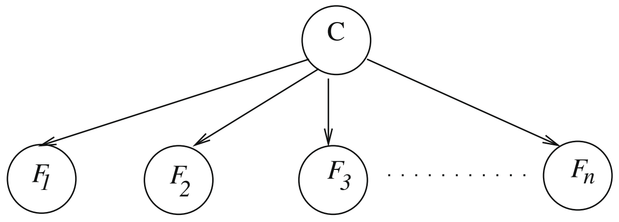

Fig. 2. A Bayesian network used as a classifier.

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 Fig. 2. Again, the independence assumption encoded by this model is that each feature is conditionally independent given the class value.

Considering Fig. 2, 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, the *posterior probability* of each class label $c \in C$, i.e. the probability of the class label given the features observed, is given by the formula:

$$P(c|f_1,\dots,f_n) = \frac{P(c)P(f_1,f_2,\dots,f_n|c)}{P(f_1,f_2,\dots,f_n)}. \tag{1}$$

Considering the independence assumption stated above, i.e. that each feature is conditionally independent given the class value, the formula reduces to:

$$P(c|f_1,\dots,f_n) = \frac{P(c)\prod_{i=1}^{n}P(f_i|c)}{P(f_1,f_2,\dots,f_n)}. \tag{2}$$

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

$$class = argmax_{c\in C}P(c)\prod_{i=1}^{n}P(f_i|c). \tag{3}$$

Notice that the denominator in Eq. (2) is a constant and can be ignored in the last step.

The proposed Bayesian Network for crawler detection has the structure shown in Fig. 2. 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 4.1. The root node represents the *class* variable.

[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.

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 Fig. 2.
- *Clicks*: Maximum sustained 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. More specifically, in our case, each feature is conditionally independent given *Class*. For example, if we know the state of *Class*, say to be a *crawler*, then a change in probability of *Images* will have no effect on the *PDF/PS* hypothesis (or any other child node hypothesis). Otherwise, if the state of *Class* is not known, then the two features are *dependent*. Any change in the probability of, say, *Images*, will cause an update in the *Class* hypothesis which will then cause a change in the probability of *PDF/PS* (and all other child nodes of *Class*). For more details on causality and learning causal structures see [32,30].

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 another in determining the classification of a session), is implicitly encoded in the conditional 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 sections we show how we use machine learning to achieve the above tasks.

4.4. Learning network parameters

The learning phase of the system uses the training data that have been created as described in Section 4.2. 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 4.3 above, and which are represented as nodes in the Bayesian network. We use the training data for variable quantization, based on the entropy, as well as for learning the conditional probability tables, as described in the next two sections.

4.4.1. Variable quantization

In this implementation, the Bayesian Network is developed for discrete variables, as classification performance tends to be better when continuous variables are discretized than when they are assumed to follow Gaussian distributions [17]. We therefore need to quantize variables 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* [29]. 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 k interval labels, the entropy is given by:

$$E = \sum_{i=1}^{k} -P_{Ci}log_2 P_{Ci} - P_{Hi}log_2 P_{Hi}, \tag{4}$$

where P_{Ci} is the probability of crawler sessions with label i and P_{Hi} is the probability of human sessions with label i. The entropy is then weighted by the fraction of examples that belong in each interval. An exhaustive search procedure was used over the number and ranges of the quantization values to determine the number and ranges of the quantization steps which minimized this entropy function. The number and range of interval labels 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 4.3: *Clicks*, *Duration*, *Images*, *PDF/PS* and *Code 4xx*.

4.4.2. 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 4.3. 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.

4.5. 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 4.1, could be (17, 135.5, 67, 2, 0, 0) which can be described as follows: the session in question had reached a peak of 17 clicks (in a pre-set 12-s window), had a session length of 135.5 s, 67% of its requests were to image files, 2% of its requests were to pdf/ps files, there were 0requests 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 4.4.1.

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.

5. 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.

For the purposes of evaluating the performance of our 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 [15]). Table 2 shows the datasets used for training and testing the system. The access logs were processed by our log analyzer to extract the sessions. We will be referring to these datasets throughout this section.

5.1. Training in the presence of class imbalance

The dataset used for training is shown in Table 2 as $S1$. Sessions were labeled using our approach described in Section 4.2. The learning stage proved to be a challenging task. The problem encountered with this stage is one of *class imbalance* [34,19,41,21,26]. 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. Some domains where the imbalanced dataset problem is present are fraud detection, network intrusion detection, cancerous cells detection etc. Because the unusual class is rare among the general population, the class distributions are very skewed [34]. The amount of imbalance varies depending on the domain: it is less than 10% for intrusion detection but less than 1% for cancerous cells detection. Our domain of crawler detection also falls in the category of domains that exhibit the imbalanced data set problem. In our earlier study [15] we have concluded that crawler activity in access logs amount to approximately 10% 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.

Resampling the training set is an approach that is often used to solve the imbalanced data set problem at the learning stage. Resampling modifies the prior probabilities of the majority and minority class by changing the records on each of the two classes. In our study we show results with and without resampling on the training data set $S1$ shown in Table 2. For resampling, we adopted two 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:

(1) *Training Data set 1*: No resampling. In this experiment, the sessions in the training data are not altered and have the prior probability as determined by the sessions extracted from the access log.
(2) *Training Data set 2*: Oversampling to 15%. Crawler sessions are randomly chosen from the original set for duplication, until they amount to 15% of the total number of sessions in the training set.
(3) *Training Data set 3*: Oversampling to 50%. Crawler sessions are randomly duplicated until they amount to 50% of the total number of sessions in the training set, i.e. until the two cases are equally represented.

Table 2
The data sets used for training and testing the system.

Data set name	Source (server)	Session length	No. of sessions	Time span	Use
S1	U. Toronto-CS/UCY-CS	$\geqslant 5$	11,094	2 months	Training
S2	UCY-CS	$\geqslant 5$	784	1 month	Testing
S3	UCY-CS	$\geqslant 5$	1804	1 month	Testing
S4	UCY-CS	$\geqslant 8$	1524	1 month	Testing
S5	U. Toronto-CS/UCY-CS	$\geqslant 5$	174	9 months	Testing
S6	gEclipse	$\geqslant 5$	315	1 month	Testing

Table 3
Training data sets used for five experiments, with and without resampling on data set *S*1 from Table 2. It includes the number of *crawler* and *human* sessions used for training in each case.

Data set no.	No. distinct humans	No. distinct crawlers	No. humans used in training	No. crawlers used in training	Prior probabilities: (human, crawler)
1	10,106	988	10,106	988	(0.91, 0.09)
2	10,106	988	10,106	1784	(0.85, 0.15)
3	10,106	988	10,106	10,106	(0.5, 0.5)
4	10,106	988	5599	988	(0.85, 0.15)
5	10,106	988	988	988	(0.5, 0.5)

(4) *Training Data set 4*: Undersampling to 85%. Human sessions are eliminated randomly until they amount to 85% of the total number of sessions in the training set.
(5) *Training Data set 5*: Undersampling to 50%. Human sessions are randomly eliminated until they amount to 50% of the total number of sessions in the training set, i.e. until the two cases are equally represented.

Table 3 shows the number of Crawler and Human sessions in each of the above training data sets, obtained via resampling data set *S*1 shown in Table 2. 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[3] (Naive Bayes 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. Throughout the remaining of the paper we will refer to the 5 classifiers as *C*1 through *C*5, with each classifier obtained using learning of the respective data set from Table 3.

5.2. Testing and evaluation

To test our classifier, we chose an access log *different* from the one used during training. More specifically, the classifiers were all tested on data set *S*2 shown in Table 2 obtained from the University of Cyprus server. This access log was processed by our log analyzer to extract the various sessions. It should be noted that we did not do any resampling on the test data. The natural imbalance in this data set was, therefore, left untouched. A human expert did an entirely manual classification of each session in the testing set in order to provide us with the ground truth by which we were to evaluate our classifier's performance. It turned out that the testing set contained 685 actual human sessions and 99 actual crawler sessions, as labeled by the independent human expert. As with training, sessions with less than 5 requests were excluded from testing. We assume that 5 is the minimum number of requests that a human expert would need in order to classify the session with confidence and provide the ground truth. We tested the performance of all five Bayesian networks (one for each data set), on the same testing dataset.

[3] The networks were implemented using the Ergo^{TM} tool [18].

5.2.1. Recall and precision

The class imbalance problem discussed above also raises issues in the evaluation of the classifier's performance. A simple evaluation based on accuracy, i.e. the percentage of correct classifications, can be misleading. To illustrate this, assume a dataset with 100 cases out of which 90 cases belong to the majority class and 10 cases belong to the minority class. Then a classifier that classifies every case as a majority class will have 90% accuracy, even though it failed to detect every single target of the minority class.

Therefore, to test the effectiveness of our classifiers, we adopted metrics that are commonly applied to imbalanced datasets: *recall*, *precision*, and the F_1-measure [39], which summarizes both *recall* and *precision* by taking their harmonic mean. F_1 summarizes the two metrics into a single value, in a way that both metrics are given equal importance. The F_1-measure penalizes a classifier that gives high recall but sacrifices precision and vice versa. For example, a classifier that classifies all examples as positive has perfect recall but very poor precision. 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 definition of these metrics follows:

$$Recall(R) = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

$$Precision(P) = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

$$F_1 = \frac{2RP}{R+P}. \tag{5}$$

Positive classification, in our study is the classification of a session as *Crawler* (the target class). The above formulae therefore translate to:

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

$$Precision = \frac{\text{No. of Crawler sessions correctly classified}}{\text{No. of predicted Crawler sessions}}. \tag{6}$$

The values of recall, precision and F_1-measure obtained by classifiers $C1, \ldots, C5$ are given in Table 4 and plotted in Fig. 3.

As it can be seen from Table 4, our crawler detection system yields promising results with both recall and precision being above 79% in all experiments performed. The lowest *F*1-measure is obtained by C1 when we train the

Table 4
Evaluation metrics of each Bayesian network classifier tested on dataset $S2$ given in Table 2. Classifiers differ in their parameters, which were obtained via training using datasets given in Table 3.

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

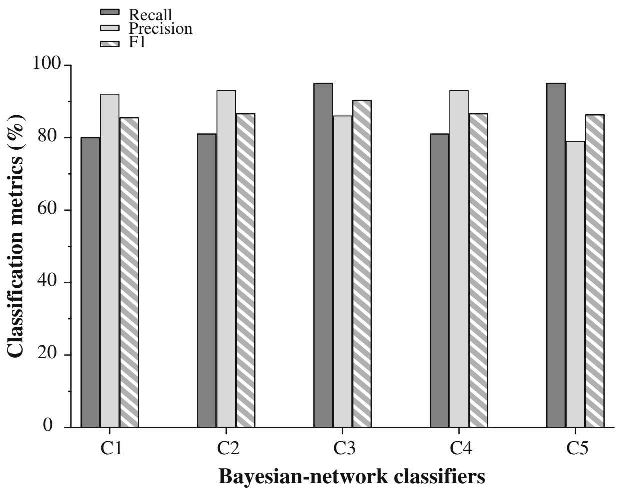

Fig. 3. An evaluation of the crawler detection classifiers.

system with 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. In the case of C5, *Humans* were randomly undersampled to reach the number of *Crawler* sessions in the original set. This meant eliminating 9118 out of the 10,106 *Humans*, a significant portion of the original set. 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. There 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.

5.2.2. *ROC Curve*

The *ROC* (Receiver Operating Characteristic) graph [34,9], can also be used to show graphically the trade-off between `true positive rate` (TPR) and `false positive rate` (FPR) of a classifier. TPR is the fraction of positive examples predicted correctly (i.e. same as *recall* defined above) whereas FPR is the fraction of negative examples predicted as positive. Formally they are defined as:

$$TPR = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$
$$FPR = \frac{\text{False positive}}{\text{True negative} + \text{False positive}}.$$

In the ROC curve the FPR is plotted along the x-axis and the TPR is plotted along the y-axis. The result of the classification is a class probability distribution given the evidence. A point on the ROC curve represents the FPR and TPR associated with the classification based on a given discrimination threshold. The threshold refers to the cut-off value above which a record is classified as positive. By varying the threshold we produce different points on the ROC curve (i.e. different (FPR,TPR) pairs). By connecting consecutive points with tracing straight lines we produce the ROC curve. A good classification model should be located as close as possible to the upper left corner of the graph, i.e. point $(0,1)$, which means close to a 0 value for FPR and value 1 for TPR. This means that the ideal model should have no negative examples predicted as positive, and all positive examples predicted correctly. Equivalently, point $(0,0)$ arises when the threshold is set such that the model predicts every instance to be negative, and point $(1,1)$ when the model predicts every instance to be a positive class. The more steeply the curve moves up and then across, the better the model. A model that makes random guesses is located along the diagonal line $y = x$. Another way to look at this is to see the *area under the curve* (AUC). A perfect model has AUC equal to 1 whereas a model that random guesses has AUC equal to 0.5. The ROC curves for each of the classifiers $C1, \ldots, C5$ are shown in Fig. 4.

5.2.3. *Majority vote*

An important conclusion that was drawn by observing the system classification of sessions, is that the sessions that shared the same IP addresses were also, in their vast majority, assigned the same classification label by the system. More specifically, we introduced a *majority vote* over all groups of sessions that shared the same IP address. This *majority vote* metric was derived by forming groups of all sessions that shared the same originating IP address and finding, for each of these groups, the percentage of sessions

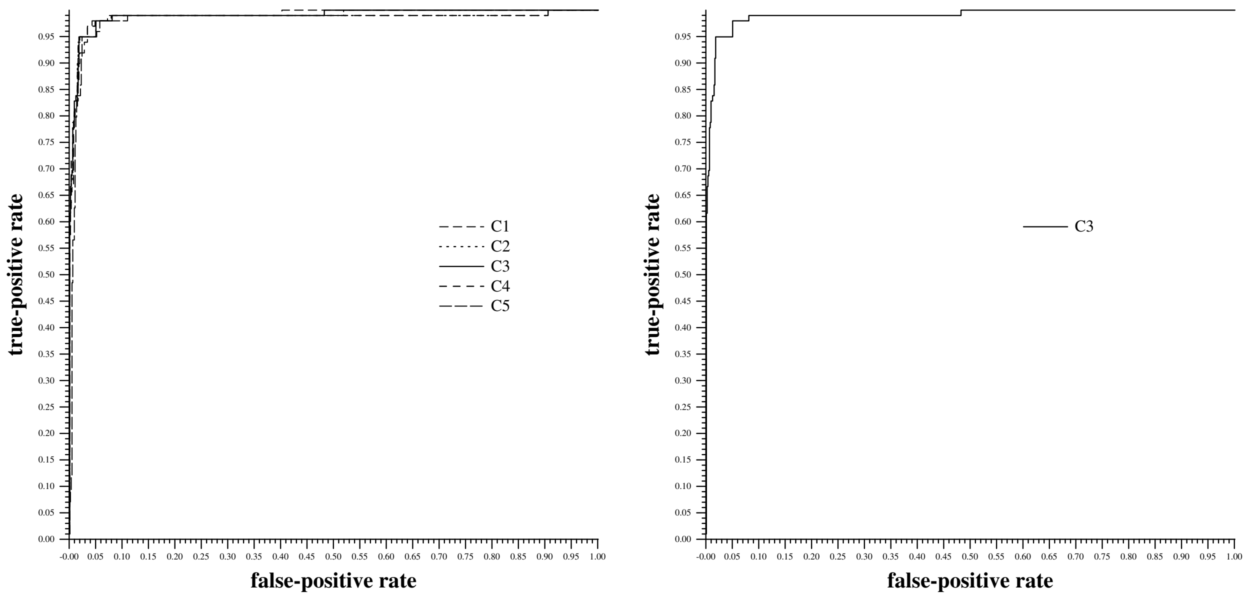

Fig. 4. ROC curves for all five classifiers (left) and for the best classifier (right).

that agreed in their classification. This percentage was then weighted by the size of the group and divided by the sum of all sessions that shared IP addresses. The *majority vote*, for our binary classification problem, ranges between [0.5, 1] and is given by:

$$\text{majority vote} = \frac{\sum_{i=1}^{k} f_i N_i}{\sum_{i=1}^{k} N_i}, \quad (7)$$

where f_i is the percentage of classification agreements in group i, N_i is the number of sessions in group i, k is the number of groups of sessions that have common IP addresses and the denominator is the sum of all sessions in each group. As it can be seen from the above metric, higher percentage agreement in large groups is more significant than agreement in smaller groups. For example, 20 sessions that share the same originating IP and have an 100% agreement in their classifications, is more significant than a 100% agreement of 3 sessions. Considering the extreme scenarios, if we have perfect agreement, i.e. f_i is 1 for all i, the *majority vote* is 1. On the other hand, a complete disagreement in which the sessions in each group are split in half, will yield a *majority vote* of 0.5.

By applying this metric to classification results obtained by our best classifier $C3$, we have a *majority vote* of 0.957. This indicates a very high agreement in classification of sessions with the same originating IP address.

The impact of this is significant as one could use this system not only to classify sessions but also to determine IP addresses that belong to crawlers and that were previously unknown. In our system we used the list of known crawler IP addresses for labeling sessions as *crawlers* in the training data only. In our classification, this information was not used as evidence and therefore all sessions were subject to classification, regardless of whether their IP belonged to a known crawler. However, we could extend our model so that the classification of the IP address (i.e. known crawler IP or not) can contribute to the class probability distribution of the session and hence, to the final session classification. The IP classification can easily be incorporated in the network as a parent node of the *Class* node, to create a general Bayesian network structure (not a Naive Bayes). In this extended network, the IP class will directly influence the session class.

5.2.4. Testing C3 on additional data

We further tested the chosen classifier $C3$ on additional unseen test data. More specifically, we tested the classifier on datasets $S3$ and $S4$ given in Table 2. It should be noted that these two datasets were obtained from the same access log but they differ in the minimum number of requests per session: $S3$ was derived by keeping only sessions with at least 5 requests, whereas $S4$ was derived by keeping only sessions with at least 8 requests. The sessions were identified as before and then labeled based on our labeling algorithm outlined in 4.2. We obtained a total of 1804 and 1524 sessions for dataset $S3$ and $S4$, respectively. The results are shown in Table 5. As it can be seen, our classifier maintains its high recall and precision of 95% and 83%, respectively, in classifying sessions with at least 5 requests. These are increased to 96% and 89% when considering only sessions with more than 7 requests. The most considerable difference comes in precision, where the number of False Positives decrease significantly by raising the number of minimum session requests from 5 to 8. This

Table 5
Evaluation of classifier $C3$ on datasets $S3$ and $S4$ given in Table 2. Both datasets come from the same access log, by varying the minimum number of requests per session.

Dataset name	No. of requests per session	No. of actual humans	No. of actual crawlers	Precision	Recall
S3	⩾5	1589	215	0.95	0.83
S4	⩾8	1323	201	0.96	0.89

indicates that a substantial number of the sessions that were wrongly classified as *Crawlers*, had between 5 and 7 requests.

5.2.5. Testing on known crawlers

In this section, we test further the accuracy of the best classifier, *C*3, presented in Section 5.2, on different test data sets consisting only of *crawler* sessions. These are datasets *S*5 and *S*6 given in Table 2. The ground truth, i.e. the actual classifications of the sessions in these data sets, were not extracted manually. Instead the sessions were given the *crawler* target classification due to the fact they had an originating IP address of a known crawler.

Starting with dataset S5, this was created by extracting all these test sessions from 4 different access log files, coming from 2 different servers residing in 2 countries. The data were not encountered during training, and they were taken in various dates. There were a total of 315 crawler sessions based on the known IP addresses. Results are shown in Table 6. Our system classified correctly 288 sessions. Out of the 27 mis-classified sessions, 18 sessions had a total of 5 requests and 3 sessions had a total of 6 requests. Examples of these mis-classifications are shown below:

Case 1:

```
[09/Nov/2001:14:38:41]"GET/courses/EPL222/exams/99f.ps HTTP/1.0" 200 772096
[09/Nov/2001:14:40:59]"GET/courses/EPL222/Slides-2/Chap8.ppt HTTP/1.0" 200 800256
[09/Nov/2001:14:41:12]"GET/courses/EPL651/Resources/papers/bbl.ps HTTP/1.0" 304 -
[09/Nov/2001:14:52:58]"GET/∼epl131/c_semester/lectures/lec2/tsld005.htm HTTP/1.0" 200 1269
[09/Nov/2001:14:57:54]"GET/courses/EPL132/notes/notes3.pdf HTTP/1.0" 304 -
[09/Nov/2001:14:57:54]"GET/courses/EPL132/notes/notes3.pdf HTTP/1.0" 304 -
```

Case 2:

```
[08/Nov/2001:07:33:10]"GET/~ epl233/JavaTutorial/info/downl.html HTTP/1.0" 200 7103
[08/Nov/2001:07:43:13]"GET/~ jorge/SAC95.ps.gz HTTP/1.0" 304 -
[08/Nov/2001:07:51:00]"GET/mdd/courses/EPL625/sqd_cache/tsld013.htm HTTP/1.0" 304 -
[08/Nov/2001:07:53:19]"GET/~ epl131/fall00/lectures/lec5/sld011.htm HTTP/1.0" 200 2355
[08/Nov/2001:07:53:41]"GET/mdd/courses/EPL625/sqd_cache/tsld005.htm HTTP/1.0" 304 -
```

Table 6
Classification accuracy using sessions with originating IP address of known crawlers.

Dataset name	No. of actual crawlers	No. of predicted crawlers	Accuracy
S5	315	288	0.914
S6	174	148	0.851

These mis-classifications produced a probability of 0.51 of the session being a *human*, and were thus assigned the wrong class compared with the target. Due to the small number of requests in these, and most of the mis-classified sessions, it is difficult even for a human expert to classify these sessions. They were given a target classification of *crawler* based solely on the known originating IP address. However, it is interesting to note that the system only marginally classified them as *human* (probability 0.51). Out of the 27 mis-classifications 24 were classified in the opposite class with weak probabilities in the range of 0.51–0.65. The maximum probability assigned to the wrong class was 0.78 and it was assigned to the three sessions shown below:

Case 3:

```
[09/Nov/2001:14:06:09]"GET/EPL224/docs/chap6_hspeed_LAN.ppt HTTP/1.0" 304 -
[09/Nov/2001:14:06:20]"GET/EPL224/docs/chap6_hspeed_LAN.ppt HTTP/1.0" 200 206848
[09/Nov/2001:14:26:03]"GET/EPL653/papers/ccontrol/nw.ppt HTTP/1.0" 304 -
[09/Nov/2001:14:26:11]"GET/EPL653/papers/ccontrol/nw.ppt HTTP/1.0" 200 112128
[09/Nov/2001:14:34:09]"GET/EPL651/Res/papers/lossprofile.ps HTTP/1.0" 200 129107
```

Case 4:

```
[08/Nov/2001:23:08:28]"GET/~ es/Gproj/Presentation/CYkpcoQ2g.doc HTTP/1.0" 304 -
[08/Nov/2001:23:08:30]"GET/~ es/Gproj/Presentation/CYkpcoQ2g.doc HTTP/1.0" 200 52736
[08/Nov/2001:23:08:49]"GET/~ mas/epl601/notes/comm.pdf HTTP/1.0" 200 163599
[08/Nov/2001:23:32:01]"GET/~ pan/EPL011-IS/Data/Proskisi.doc HTTP/1.0" 304 -
[08/Nov/2001:23:32:03]"GET/~ pan/EPL011-IS/Data/Proskisi.doc HTTP/1.0" 200 27136
```

Case 5:

```
[08/Nov/2001:21:23:48]"GET/cgi-bin/netforum/epl221/a/7–7 HTTP/1.0" 200 1942
[08/Nov/2001:21:26:18]"GET/courses/EPL425/notes/slides4intro.pdf HTTP/1.0" 304 -
[08/Nov/2001:21:44:41]"GET/courses/EPL224/exams/final98.doc HTTP/1.0" 200 113664
[08/Nov/2001:21:49:04]"GET/courses/EPL224/exams/final00.doc HTTP/1.0" 304 -
[08/Nov/2001:21:49:07]"GET/courses/EPL224/exams/final00.doc HTTP/1.0" 200 93184
```

The mis-classified sessions shown above, even though they have a small number of requests as did Cases 1 and 2, they were more "confident" in their wrong classification (probability 0.78 as opposed to 0.51). The reason is that they contain requests for files such as *.doc* or *.ppt* and not for *html*.

Dataset *S*6 is a very recent log file from the site of an open-source software project, covering period August

2007 until end of May 2008. With this log we wanted to test our classifier with the recent trends and observe whether our network, which was trained with an older set of data, was still effective in detecting crawlers. As it can be seen from 6, the accuracy (i.e. the percentage of crawler sessions classified correctly) when testing with the recent log, *S*6, is 85% compared to the 91.4% obtained using the older data set *S*5. We consider the new results to be equally promising considering that the training of the system used older sets of data, of the same period as *S*5. The training reflects the current trends in crawler techniques and the fact that the classifier is still quite accurate shows that the main trends as exhibited by the features we used in our network, have not changed dramatically the last 6 years. However, after investigating the mis-classified sessions further, we found one recent trend that was not present in our older data, and that is the dynamic generation of web content. All mis-classified sessions of dataset *S*6 contained PHP requests, which were considered for determining the maximum sustained click rate in *testing* with the recent data but were not included in the *training* of the network. The system was not taught with examples of this type of sessions and therefore its parameters on the click rate feature could use some further tuning to reflect this current trend. Regardless, its impact on the overall accuracy was not so dramatic. Our system remains relevant and effective for detecting crawlers.

6. Conclusions and future work

In this paper we presented the use of a probabilistic model, namely a Bayesian network, for detecting crawlers inside Web-server 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.

During our training and testing stages we also addressed the class imbalance problem that arises when the class distributions in data sets are highly skewed. We have used resampling to counter this 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. Our recall and precision were consistently above 95% and 83% reaching as high as 96% and 89%, respectively if we ignore sessions with less than 8 total requests. Moreover, our proposed "majority vote" metric shows a high agreement in classification which indicates that our system can be extended to determine IP addresses that belong to crawlers, which were previously unknown.

The 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 investigate the effectiveness of other features of Web sessions, such as the navigational semantics of user-agent requests.

Acknowledgement

The authors wish to thank Andoena Balla for her help with collecting and processing the gEclipse logs.

References

[1] Google Advertising. <http://www.google.com>/ads (last accessed November 2005).

[2] Search engine robots. <http://www.jafsoft.com>/searchengines/webbots.html (last accessed October 2005).

[3] Search Engine Watch: Tips about Internet Search Engines and Search Engine Submission. <http://searchenginewatch.com>/ (last accessed October 2005).

[4] The Web Robots Pages. <http://www.robotstxt.org>/ (last accessed October 2005).

[5] Google and Yahoo! accused of click fraud collusion. ElectricNews.Net, April 5, 2005. <http://www.theregister.co.uk>.

[6] Referer spam. Wikipedia, the free encyclopedia, August 2007. <http://en.wikipedia.org>/wiki/Referer_spam.

[7] V. Almeida, D. Menascé, R. Riedi, F. Peligrinelli, R.Fonseca, W. Meira Jr, Analyzing web robots and their impact on caching, in: Proceedings of the Sixth International Workshop on Web Caching and Content Distribution, June 2001, pp. 299–310.

[8] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, Searching the Web, ACM Transactions on Internet Technology 1 (1) (2001) 2–43.

[9] G. Batista, R. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explorations 6 (1) (2004) 20–29.

[10] J. Bilmes, Dynamic Bayesian multinets, in: Craig Boutilier, Moisés Goldszmidt (Eds.), UAI'00: Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, 2000, pp. 38–45.

[11] S. Brin, L. Page, The anatomy of a large-scale hypertextual (web) search engine, Computer Networks and ISDN Systems 30 (1–7) (1998) 107–117.

[12] S. Chakrabarti, M. van den Berg, B. Dom, Focused crawling: a new approach to topic-specific web resource discovery, in: Proceedings of the 8th World Wide Web Conference, Elsevier Science, Toronto, 1999, pp. 545–562.

[13] K. Crawford, Google CFO: Fraud a big threat. Google exec calls click fraud the "biggest threat" to the Internet economy, urges quick action, CNN Money, December 2, 2004. <http://www.cnn.com>.

[14] M.D. Dikaiakos, A. Stassopoulou, L. Papageorgiou. Characterizing crawler behavior from Web-server access logs, in: A. Min Tjoa, K. Bauknecht, G. Quirchmayr (Eds.), Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003), Lecture Notes in Computer Science, vol. 2738, Springer, September 2003, pp. 369–378.

[15] M.D. Dikaiakos, A. Stassopoulou, L. Papageorgiou, An investigation of WWW crawler behavior: characterization and metrics, Computer Communications 28 (8) (2005) 880–897.

[16] M.D. Dikaiakos, D. Zeinalipour-Yazti, A distributed middleware infrastructure for personalized services, Computer Communications 27 (15) (2004) 1464–1480.

[17] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Proceedings of the Twelfth International Conference on Machine Learning, 1995, pp. 194–202.

[18] Noetic Systems Incorporated. <http://www.noeticsystems.com/ergo/index.shtml>.

[19] Andrew Estabrooks, Taeho Jo, Nathalie Japkowicz, A multiple resampling method for learning from imbalanced data sets, Computational Intelligence 20 (1) (2004) 18–36.

[20] Severino F. Galán, Francisco Aguado, Francisco Javier Díez, José Mira, Nasonet, modeling the spread of nasopharyngeal cancer with networks of probabilistic events in discrete time, Artificial Intelligence in Medicine 25 (3) (2002) 247–264.

[21] N. Japkowicz, Learning from imbalanced data sets: a comparison of various strategies, in: Learning from Imbalanced Data Sets, Papers from the AAAI Workshop, Technical Report WS-00-05, 2000, pp. 10–15.

[22] R.J. Kennett, K.B. Korb, A.E. Nicholson, Seabreeze prediction using Bayesian networks, in: David Wai-Lok Cheung, Graham J. Williams, Qing Li (Eds.), Knowledge Discovery and Data Mining – PAKDD 2001, 5th Pacific-Asia Conference, Lecture Notes in Computer Science, vol. 2035, Springer, 2001, pp. 148–153.

[23] R. Kohavi, Mining e-commerce data: the good, the ugly and the bad, in: KDD'01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, NY, USA, 2001, pp. 8–13.

[24] K. Komando, Here's how to make blogging pay, USA Today, September 18, 2005. <http://usatoday.com>.

[25] B. Krishnamurthy, J. Rexford, Web Protocols and Practice, Addison-Wesley, 2001.

[26] M. Kubat, R.C. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Machine Learning 30 (2–3) (1998) 195–215.

[27] S. Mani, S. McDermott, M. Valtorta, Mentor: A Bayesian model for prediction of mental retardation in newborns, Research In Developmental Disabilities 18 (5) (1997) 303–318.

[28] D.A. Menasce, V. Almeida, R. Fonseca, R. Riedi, F. Ribeiro, W. Meira, In search of invariants for e-business workloads, in: 2000 ACM Conference In Electronic Commerce, ACM, 2000, pp. 56–65.

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

[30] Richard E. Neapolitan, Learning Bayesian Networks, Pearson Prentice Hall, 2004.

[31] Judea Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufman Publishers Inc., 1988.

[32] Judea Pearl, Causality: Models Reasoning and Inference, Cambridge University Press, 2000.

[33] Thang V. Pham, Marcel Worring, Arnold W.M. Smeulders, Face detection by aggregated Bayesian network classifiers, Pattern Recognition Letters 23 (4) (2002) 451–461.

[34] Foster J. Provost, Tom 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.

[35] J. Srivastava, R. Cooley, M. Deshpande, P.-N. Tan, Web usage mining: discovery and applications of usage patterns from web data, SIGKDD Explorations 1 (2) (2000) 12–23.

[36] A. Stassopoulou, T. Caelli, Building detection using Bayesian networks, International Journal of Pattern Recognition and Artificial Intelligence 14 (6) (2000) 715–733.

[37] A. Stassopoulou, M. Petrou, J. Kittler, Application of a Bayesian network in a GIS based decision making system, International Journal of Geographical Information Science 12 (1) (1998) 23–45.

[38] Pang-Ning Tan, Vipin Kumar, Discovery of web robot sessions based on their navigational patterns, Data Mining and Knowledge Discovery 6 (1) (2002) 9–35.

[39] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, Addison-Wesley, 2005.

[40] 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.

[41] Gary M. Weiss, Mining with rarity: a unifying framework, SIGKDD Explorations 6 (1) (2004) 7–19.

[42] S. Ye, G. Lu, X. Li, Workload-aware web crawling and server workload detection, in: Network Research Workshop, 18th Asian Pacific Advanced Network Meeting (APAN 2004), July 2004. <www.cn.apan.net>/cairns/NRW/ (last accessed, December 2004).

Dr. Athena Stassopoulou is currently a Professor of Computer Science at the University of Nicosia. She holds a B.Sc. in Computer Science and Mathematics (joint honors) from the University of Manchester, UK (1992) and a PhD in Artificial Intelligence, from the University of Surrey, Centre for Vision Speech and Signal Processing, UK (1996). She worked as a Post-doctoral Researcher and as a Senior Research Associate at the Center for Mapping (NASA Commercial Space Center), The Ohio State University, USA. She has more than 15 years of research experience and has worked in projects funded by NASA (Image Understanding Initiative), the National Imagery and Mapping Agency (NIMA) in the USA, the European Union and the Research Promotion Foundation of Cyprus. She has also served as an External Evaluator and a Scientific Reviewer in projects of the Information Society and Technology (IST) Programme of the European Union. Dr. Stassopoulous research interests are in: uncertain reasoning, Bayesian Networks, Machine Learning, Computer Vision and Image Understanding and more recently, in Artificial Intelligence Applications for the Web.

Marios D. Dikaiakos is an Associate Professor of Computer Science at the University of Cyprus, where he established and leads the High-Performance Computing Systems Laboratory. Dikaiakos received his Ph.D. from Princeton University (1994), an M.A. degree from Princeton (1991), and a Dipl.-Ing. degree from the National Technical University of Athens (summa cum laude, 1988). He was a Research Associate at the University of Washington in Seattle (1994–1995) and spent a pre-doctoral internship at the Paris Research Lab of Digital Equipment Corporation (fall 1991). He has been a Socrates/Erasmus visiting professor at the University of Crete and ICS FORTH (10/2004), and a visiting research professor at the Department of Computer Science, Rutgers University (Spring 2005). Research and Teaching Activities Dikaiakos' research focuses on network-centric computing, with an emphasis on Grids, Vehicular Ad-Hoc Networks, and the World-Wide Web. Dikaiakos has been a principal investigator for several projects funded by the European Union and the Research Promotion Foundation of Cyprus.