Web robot detection techniques: Overview and limitations

Article *in* Data Mining and Knowledge Discovery · June 2011

DOI: 10.1007/s10618-010-0180-z · Source: DBLP

Derek Doran
Wright State University
103 PUBLICATIONS **1,141** CITATIONS

Data Min Knowl Disc
DOI 10.1007/s10618-010-0180-z

Web robot detection techniques: overview and limitations

Derek Doran · Swapna S. Gokhale

Received: 1 June 2009 / Accepted: 16 June 2010
The Author(s) 2010

Abstract Most modern Web robots that crawl the Internet to support value-added services and technologies possess sophisticated data collection and analysis capabilities. Some of these robots, however, may be ill-behaved or malicious, and hence, may impose a significant strain on a Web server. It is thus necessary to detect Web robots in order to block undesirable ones from accessing the server. Such detection is also essential to ensure that the robot traffic is considered appropriately in the performance and capacity planning of Web servers. Despite a variety of Web robot detection techniques, there is no consensus regarding a single technique, or even a specific "type" of technique, that performs well in practice. Therefore, to aid in the development of a practically applicable robot detection technique, this survey presents a critical analysis and comparison of the prevalent detection approaches. We propose a framework to classify the existing detection techniques into four categories based on their underlying detection philosophy. We compare the different classes to gain insights into those characteristics that make up an effective robot detection scheme. Finally, we discuss why the contemporary techniques fail to offer a general solution to the robot detection problem and propose a set of key ingredients necessary for strong Web robot detection.

Keywords Web Crawler · Web Robot · WWW · Web Robot Detection · Web User Classification

Responsible editor: Charles Elkan.

D. Doran (✉) · S. S. Gokhale
Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, USA
e-mail: derek.doran@engr.uconn.edu

S. S. Gokhale
e-mail: ssg@engr.uconn.edu

1 Introduction

The World Wide Web (WWW) has evolved from being a massive information repository to a new medium for communication and collaboration. This evolution is powered by next generation Web applications which provide novel, value-added services. Web robots form a critical component of many of these Web applications and services. Research in Web robots has also contributed to the development of powerful search engines which make the Internet usable for everyone across the world. Web robots may be generally defined as autonomous systems that send requests to Web servers across the Internet to request resources. The robots then analyze the results received to gain some knowledge, which is then used to fulfill some specific purpose in a broader context. A canonical example of a Web robot is a search engine indexer, while a less common example is an RSS feed crawler or a robot designed to collect Web sites for an Internet archive.

With a sudden surge in second generation Web applications centered around Web-based communities and hosted services (i.e., social networking sites, wikis, podcasts, user blogs, and RSS feeds) dubbed *Web 2.0* (Oriley 2007), there has been a significant increase in the volume of time-sensitive, dynamic, and current information that is posted to the Internet. For data repositories to stay up-to-date with the most recent and relevant available information, contemporary Web robots must be more comprehensive in their searches, more specialized in their functionality (to analyze a wider variety of data), and more frequent in their visits (to stay abreast on content that changes dynamically and rapidly). As a result, there has been a significant rise in the number and types of robots as well as their aggressiveness when they crawl Web servers. As an example, Lee et al. found that the eight of the most active Web crawlers on Microsoft Web servers widely vary in the types of resources requested as well as in the size of the resources they request (Lee et al. 2009).

The technical know how to build sophisticated robots can be abused to build "bad" robots in the wild. Examples of such bad robots include those that may try to download Web sites for malicious purposes such as building copycat sites, generating spam, or to act unethically (Giles et al. 2010). Other "bad" robots show behavior that is not becoming to their intended functionality as a result of flawed engineering. An example is a link verifier that issues a GET command for resources rather than sending a HEAD http request to check if the file is available. Intentionally or accidentally, "bad" robots often go undetected and unnecessarily drain server resources. Bad robots must therefore be detected and blocked from accessing and utilizing server resources.

The growth of the Internet has also led to an inevitable growth in the traffic from Web robots. In an analysis of the Web logs generated from the University of Connecticut (UConn) School of Engineering (SoE) Web server (Doran et al. 2009) over a year-long period from February 2007 through January 2008, we observed that 18.5% of all http requests were sent by Web robots, representing 3.52% of all bytes transferred. Another recent analysis of Web robot traffic by Huntington et al. found that at least 32.6% of all traffic seen by an online scientific journal site was from Web robots (Huntington et al. 2008). By comparison, an analysis of access logs by Dikaiakos et al. in 2004 found that robots from five major search engines represented 8.51% of all http requests and accounted for only 0.65% of all bytes transferred

(Dikaiakos et al. 2005). The increasing trend in the traffic induced by Web robots is evident from a comparison of these studies. Moreover, we believe that Web robot traffic will continue to grow as online services evolve and the promise of semantic Web technologies is realized. Therefore, given the forecasted trend towards increasing robot traffic, it is imperative that this traffic be factored into the performance analysis and capacity planning of Web servers, in order to ensure that human visitors will experience a superior quality of service, despite the presence of robots. Accounting for such robot traffic requires a strong characterization of favorable Web robot activity, which in turn requires a technique that separates Web robots from all other Web server traffic.

In the early approaches designed to discover robots from Web logs, the user-agent field of each http request in the logs was compared against a database of regular expressions. The contents of this database, however, cannot be updated at the same pace as the rapid growth in the novelty and sophistication of Web robots. As a result, referencing a static database can only be useful in detecting old and common Web robots. Thus, while this method may be simple to implement and sufficient to detect common robots, it may not be as effective in detecting new and evolving robots. A limited number of research efforts over the past decade have refined these simple detection approaches, enriching the ability to detect a broader range of robots. These techniques vary widely in their underpinning detection philosophy. For example, some consider different fields in the Web log or identify simple patterns in sessions, while others perform a statistical analysis across different features of Web robot traffic or use machine learning models for detection. Some techniques even implement an entire system to detect robots in real time, rather than by mining features in the Web server access logs. While each of these techniques has been effective in its limited experimental demonstration, their critical comparison either via experimentation or by a reasoning of their features and capabilities is lacking. We argue that in order to develop robust robot detection approaches that will continue to remain effective as the robot traffic evolves, it is first necessary to understand these contemporary techniques and identify their strengths, weaknesses, and differences.

In this paper, we propose a classification scheme to categorize contemporary Web robot detection techniques. The purpose of the scheme is to exploit the commonalities across the different techniques and to highlight their differences. Under this scheme, the techniques are classified into four categories depending on their underlying detection philosophy. We then provide a brief overview of the techniques belonging to each category. Subsequently, we compare and contrast the main features of these categories to identify the key ingredients that may make up an effective detection strategy. Based on the comparative analysis, we propose a new direction in Web robot detection research that incorporates fundamental distinctions between Web robot and human traffic in an analytical machine learning model.

The layout of the paper is as follows: Section 2 formulates the robot detection problem. Section 3 introduces the classification scheme and discusses the characteristics and techniques in each class. Section 4 compares and contrasts the different classes of techniques, discussing their limitations. Section 5 discusses how these limitations may be addressed by a new detection approach. Concluding remarks are offered in Sect. 6.

2 Robot detection problem

We define the problem of identifying robot traffic on a server as the Web robot detection problem, and formalize it as follows: Given a set of http request records R, for each record $r \in$ R classify r with a label specifying whether the request was sent by a human user or sent by some automated program or a system. Examples of such programs or systems include Web site indexers, spambots, and utility programs that scan Web sites for security vulnerabilities. Each r follows a format defined by the Web server administrator, depending on how verbose or simple they wish the server access log to be. Traditionally, r consists of at least the address from which the request originated, the location and the name of the resource requested, the http response code (i.e., 404 if the file is not found on the server), and the user-agent field from the request packet.

Many proposed solutions to the robot detection problem classify *sessions* rather than individual records. A session $\mathbf{S} \subseteq$ R is a collection of all request records from a user during a single visit. In this case, the label assigned to $\mathbf{S}$ is assigned to each $r \in \mathbf{S}$ implicity, so that all $r \in$ R are classified. For techniques that characterize sessions instead of individual requests, it is also necessary to identify such sessions from the collection of records. We formalize the session identification problem as follows: Given a set of http records R, find the set of sessions $\mathcal{S} = \{\mathbf{S_1}, ..., \mathbf{S_n}\}$ such that the sessions of $\mathcal{S}$ are mutually exclusive and $\bigcup_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{S_i} = \text{R}$.

Most detection techniques discover robot sessions in a Web server access log that records an entry for each http request received. Such analysis detects a robot after it has crawled a site, and hence, we term it as *offline robot detection*. Some techniques, however, aim to detect robots in real time, while the robot is still crawling the site. Such real-time detection techniques do not operate over Web server logs, and hence, for these techniques the above formalization of the robot detection problem is inappropriate. These techniques instead solve a variant called the *real-time robot detection problem*, which we formulate as follows: given a set of active sessions on a Web server $\mathcal{S}_{\mathcal{A}}$, determine whether the traffic in each session $\mathbf{S} \in \mathcal{S}_{\mathcal{A}}$ is produced by a human or by a robot before the session $\mathbf{S}$ is terminated. This variant analyzes an incoming stream of requests to determine in real time whether an active session is human or robot-induced. Naturally, real-time robot detection is more difficult than detection based on server access logs as the data used for detection is limited only to observations from the current session.

The real-time robot detection problem may be thought of as a combination of both *intrusion prevention* and *intrusion detection* problems. These are related approaches that attempt to prevent unauthorized users from accessing a given system (the prevention problem) or detecting that unauthorized users are currently using the system (the detection problem). The main advantage of real-time robot detection is that it makes it possible to impose immediate restrictions on the robots' access to site resources. Such restrictions might involve blocking the robots, preventing them from accessing specific resources, or servicing their requests with low priority compared to human sessions when the server is under excessive load. The margin of error in real-time detection must be very low, however, to minimize the risk of alienating a human because of an incorrect classification and the subsequent imposition of restrictions. Real-time robot detection is a hybrid of both intrusion prevention and detection aimed specifically at Web robots, where their visits may be detected before or during an active crawl of the site.

Solutions to each of the above robot detection problems are important in different contexts. Real-time detection is most important when the security of a site is of paramount concern. As already alluded to, sensing what active sessions are from robots can help halt these robots from continuing to browse a site and from accessing sensitive materials. In addition, real-time detection could be used to block robot attacks by terminating their sessions before a server-side exploit or DDoS attack causes any damage. Solutions to the offline detection problem are useful to filter robot traffic for a characterization study that considers only humans or vice versa. We classify and summarize the existing techniques that solve both types of detection problems in the next section.

3 Classification scheme

We classify the existing robot detection techniques into four categories, according to their underpinning analysis or detection philosophy. We first describe the four categories and then summarize the techniques belonging to each one of them. The first three categories include techniques that solve the offline detection problem, while the fourth category is comprised of techniques that solve the real-time detection problem. The categories are presented in the order of their detection "strength". We define the strength of a detection category as its potential to detect previously unknown but well-behaved robots or sophisticated robots designed to evade detection. In practice, the selection of a specific type of technique used should balance the complexity and feasibility of its implementation against protecting the security of the site. For example, if the goal of detection is to extract a sample of Web robots from a Web log or to filter common, high-demand search engine Web robots such as *Googlebot* or *MSNBot* from an access log before running usage reports, a weak detection technique that is simple to implement may be appropriate. If the goal of detection, however, is to block robots that mask their visits, or to identify specific types of robots that may pose a security risk to a site, a stronger technique may be suitable. The four types of Web robot detection techniques are as follows:

- **Syntactical log analysis:** Syntactical log analysis techniques attempt to discover robots through a simple processing of access logs. This processing involves looking for keywords in user-agent fields of robot requests or reviewing the host addresses to identify locations from where robots are known to originate. Some of the earlier detection techniques were based on syntactical log analysis of server access files. The primary appeal of these techniques lies in the simplicity of their implementation and their reliance on a readily available file. Unfortunately, such techniques can only find those robots that are previously known, because they are based on a prior knowledge of specific key words and addresses from which robots originate.

- **Traffic pattern analysis:** Traffic-based analysis techniques seek to find common characteristics of Web robot traffic that contrast with the features of human traffic.

For example, a traffic analysis technique may rely on discovering conventional robot navigational patterns that involve a DFS or a BFS search of all embedded links and resources from a main, home, or an index page of a Web site. Another technique may analyze the types of resources that robots request to discover patterns indicative of robot behavior, or consider the statistical contrasts between robot inter-arrival times or session length distributions with those of a human visitor. The key feature of traffic pattern analysis techniques is that they attempt to detect robots based on fixed expectations about their behavior rather than using the knowledge of their names and origins as in the case of syntactical processing techniques. Thus, we consider traffic analysis based detection as stronger than detection based on syntactical log analysis.

– **Analytical learning techniques:** The random nature of robot visits and Web server traffic naturally lead to formal probabilistic models as a way to detect robots. These techniques exploit the observed characteristics of the logged sessions to estimate the likelihood that a given session was generated by a robot using a formal machine learning or a probabilistic model. The sessions may be characterized using several metrics such as time between requests, session length, html-to-image request ratio, and percentage of requests made using the http HEAD command. Analytical learning techniques can also be retrained over time as the statistical trends in Web robot traffic evolve. Thus, we consider analytical learning techniques to be even stronger than detection based on traffic pattern analysis. The primary disadvantage of analytical learning techniques, however, is that they require a training data set that includes robots that are difficult to detect in order to ensure reasonable detection accuracy.

– **Turing test systems:** Analogous to how a Turing test tries to reliably classify a conversation as being produced by a computer or a human in real time (Turing 1950), the robot detection problem tries to determine if server sessions are being produced by a robot or a human, where the http request/response pairs within a session is like a dialogue. We classify a technique that issues a test to active sessions, analyzes the response, and then classifies the session based on the results as a Turing test system. Such systems are designed to perform real-time Web robot detection, a type of intrusion detection problem. These tests may either present a CAPTCHA (Ahn et al. 2003) challenge to the visitor or engineer Web sites such that the occurrence or non occurrence of a certain event can help to determine whether the session passes or fails the Turing test. While these techniques have a theoretical and algorithmic underpinning, they are best recognized by their use of creative engineering techniques to force active sessions to take the Turing test. In the context of Web robot detection, active sessions that fail the Turing test can immediately be classified as Web robots.

Figure 1 visualizes our classification scheme for Web robot detection techniques. As previously discussed, the techniques may be first classified according to whether they perform offline or real-time detection. Real-time detection techniques are labeled as Turing test systems. Offline detection techniques may be further refined into syntactical

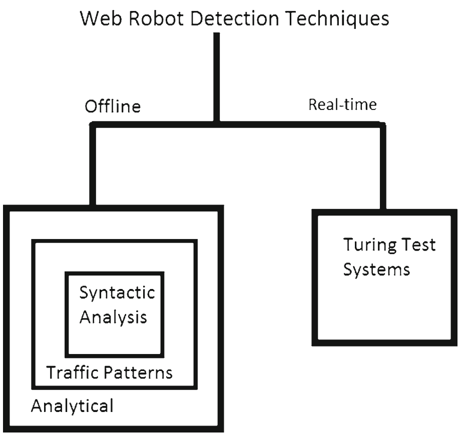

Fig. 1 A hierarchy of Web robot detection techniques. Techniques falling into a nested class are more limited in the number of Web robots they may detect

log analysis, traffic pattern analysis, and analytical learning models. The nested structure in Fig. 1 reflects the increasing strength of each type of technique in the offline detection category. Techniques that fall into a nested classification are more limited in the types of Web robots that they may detect. Table 1 summarizes the major techniques that belong to each category that this survey covers. Next, we discuss each of these techniques in detail.

3.1 Syntactic log analysis techniques

Syntactic log processing techniques analyze each entry in the server access log based on the information recorded in the log. The first technique in this category considers only individual fields in the logged request. The second technique references a database of known robot user-agent fields, and subsequently compares each request against this database. The final technique identifies robots based on multiple facets recorded in a Web server log.

3.1.1 Detection through individual field parsing

The most simple syntactic detection techniques commonly used in many commercial and open-source systems, such as *AWStats* (AWStats 2008), rely on comparing individual fields of each request against a database or a list of keywords that indicate that the request originates from a Web robot. While very simple to implement and understand, robots can easily circumvent detection by setting such fields to some arbitrary strings or by copying the fields for some well-known user agents. This is a rampant problem—in our study of Web robot activity from the UConn SoE Web server, *AWStats* detected a disproportionate volume of http requests from the three most popular robots on the Internet—*Yahoo Slurp, MSNBot*, and *Googlebot* (Doran et al. 2008). We believe that such a weak technique is still used in popular commercial systems because most administrators still do not consider robot traffic as a significant cause for concern despite evidence to the contrary.

Table 1 Overview of the major Web robot detection techniques reviewed in this survey

Type	Technique	Characteristics
Syntactic	Individual field parsing (Sect. 3.1.1)	Compares fields against database of known robots
Syntactic	User-agent field parsing (Sect. 3.1.2)	Classifies visitors based on user-agent field
Syntactic	Multi-facet analysis (Sect. 3.1.3)	Considers IP address, *robots.txt*, and user-agent field
Pattern	Combining syntactic and basic pattern analysis (Sect. 3.2.1)	Considers log parsing and simple patterns
Pattern	Resource request patterns (Sect. 3.2.2)	Considers the types of resources requested
Pattern	Query rate pattern analysis (Sect. 3.2.3)	Analyzes Web search logs based on query rates
Pattern	Novel traffic metrics (Sect. 3.2.4)	Defines novel traffic metrics for detecting Web robots
Analytic	Decision Tree (Sect. 3.3.1)	Trains decision tree using a feature vector
Analytic	Neural Network (Sect. 3.3.2)	Trains a neural net using a feature vector
Analytic	Bayesian Network (Sect. 3.3.3)	Trains a naive Bayesian network using a feature vector
Analytic	Hidden Markov Model (Sect. 3.3.4)	Uses an HMM model based on time between requests (batch arrivals)
Turing	CAPTCHA (Sect. 3.4.1)	Challenges user to solve a puzzle before getting access to a requested resource
Turing	Implicit patterns of humans (Sect. 3.4.2)	Javascript-based test automatically taken
Turing	Kill-Bots (Sect. 3.4.3)	Access control mechanism used during a DDoS attack by robots

3.1.2 Detection through user-agent mapping

Kabe and Miyazaki (2000) classify clients based on the specific application being used, rather than classifying the sessions as human- or robot-induced. They argue that learning the specific application type is important to ensure that a Web page is properly presented and to protect the Web site from abuse. Their classification system partitions the types of user agents into browsers, indexing robots, offline browsers, link checkers (a type of robot), update detectors (applications that poll a site to notify a user if there is an update), accelerators (pre-fetchers that get a target of a hyperlink contained in the current document before the user selects it), cache servers, and BBS Autopilot (a tool that checks and posts information or spams advertising on Web forms).

The analysis is applied to the access logs using a two-stage approach. In the first stage, the user-agent strings retrieved from the access logs are classified according to the product. Similar user-agent strings that resemble the same product (i.e., Mozillia 4.0, MSIE/5, etc.) are aggregated into a single product name using predefined rules. In the second stage, rules are developed to map each product name to a user-agent characteristic (Browser, Offline Browser, Indexing Robot, etc.). For example, the user-agent string `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; MathPlayer 2.1;.NET CLR 1.1.4322)` represents a visit by Internet Explorer 6.0 with MathPlayer 2.1 installed. In stage one, this user-agent field would be aggregated into the Internet Explorer product name. In stage two, the Internet Explorer product name would be aggregated into the Browser characteristic class.

The results indicate that the technique detects 2% more offline browsers (a type of harvester robot) that would be undetected by a weaker http-based method (using first token of the user-agent string). This work was among the earliest that gave a scheme to detect unique types of robots, such as spam harvesters and logo trackers (Prince et al. 2005).

3.1.3 Detection through multifaceted log analysis

Huntington et al. proposed a multi-step log analysis technique which utilizes syntactic parsing to detect robots that visit the online scientific journal *Glycobiology* (Huntington et al. 2008). In the first step, the IP addresses that sent a request for *robots.txt* were identified and immediately classified as robots. The robots identified in this step accounted for 0.5% of all traffic. Next, a reverse DNS lookup was performed for each IP address in the log file. The DNS names that included the terms robot, bot, search, spider, and crawler were checked by a human expert to be manually classified as Web robots. The robots identified by the reverse DNS lookup accounted for 16.1% of all traffic. Finally, the authors used a database of IP addresses of known robots, that do not declare themselves in their user-agent fields. This IP address database was obtained from the Web. The robots identified using this IP address database accounted for an additional 6.5% of traffic. In total, they found that their multi-faced syntactic log analysis identified robots that accounted for almost a third (32.6%) of all traffic. The percentage of robots at this site may be high because only a small number of human users are interested in visiting an online scientific journal, whereas archival or scholarly article search services will commonly employ robots to visit the journal frequently to index or archive new articles.

3.2 Traffic pattern analysis techniques

Detection techniques which analyze patterns in robot traffic use a deeper interpretation of the entries in the Web server access log rather than superficially focusing on the contents of their user-agent fields. They consider traffic characteristics such as the types of resources requested and attributes of requests such as the volume, referring location, percentage of errors, and time-of-day. In establishing these detection techniques, the authors first study Web robot traffic to identify distinguishing characteristics and statistical patterns. The detection technique is then formulated to automatically discover these distinguishing features in order to classify clients as humans or Web robots.

3.2.1 Detection through syntactic and pattern analysis

Geens et al. propose a detection technique which combines syntactic analysis with traffic pattern analysis (Geens et al. 2006). They first study the detection performance of several different syntactic parsing methods individually. The first parsing technique considers requests to the file `robots.txt`. This file is defined in the Robot Exclusion Standard (Koster 1994), that proposes a protocol where Web robots request `robots.txt` when they first visit a site. They reason that this is an unreliable approach to detect robots because this protocol is voluntary, and hence unenforceable. Next, they compare the IP address field of the request against a list of addresses from which Web robots are known to originate. The large number of robots, combined with the presence of proxy servers and dynamic IP addresses, lead them to conclude that this also is an unreliable approach to detect robots. Finally, they compare the user-agent field against the entries in a database of known regular expressions.

These individual parsing techniques were evaluated on the set of 241 Web robots that were identified manually from a Web access log. The performance of each technique was determined using the metrics *recall* (r) and *precision* (p) defined as:

$$r = \frac{|\mathbf{S}_r^c|}{|\mathbf{S}_r|}$$
$$p = \frac{|\mathbf{S}_r^c|}{|\mathbf{S}_r^p|}$$

where $\mathbf{S}_r$ is the set of all robot sessions, $\mathbf{S}_r^p$ is set of all sessions labeled as a robot by the detector, and $\mathbf{S}_r^c \subseteq \mathbf{S}_r^p$ is the set of all sessions labeled correctly. r measures the percentage of robots captured while p measures the *precision*, or percentage of sessions correctly classified as robot induced. Table 2 summarizes the results, which suggests that each individual field parsing techniques lead to very few false positives at the cost of missing a significant number of Web robots.

In order to improve r, the percentage of robots detected, they also consider three simple patterns in their traffic. The first identifies requests that use the http HEAD command to retrieve only the headers of the http responses rather than the entire response. They choose this feature because they reason that some Web robots will use the HEAD command if they are only interested in checking whether a certain resource exists, such as link checkers, while humans using a Web browser will use the http GET command to retrieve an entire html page from the server. The next pattern checks whether all requests in a session have an unassigned referrer field. Most robots

Table 2 Performance of individual field parsing approaches (Geens et al. 2006)

Approach	Correct	Incorrect	Recall	Precision
manual	241	0	1	1
robots.txt	41	0	.1701	1
ip address	167	0	.6929	.9940
user-agent	64	0	.2656	1

choose to leave the referring field blank, while Web browsers often assign the previous page visited to the referrer field. Finally, the authors examine whether any image files are requested during a session. They consider image files because most robots are not interested in the embedded images on html pages.

This combined approach leads to the following rule to detect Web robots: *if a session contains a request for `robots.txt` OR its IP address is in the robot IP list OR its user-agent field is in the robot user-agent list OR the HEAD method is used OR (the referring field is unassigned AND no images are requested) then classify the session as a Web robot*. This rule obtains a recall value of 0.9731 with precision 0.8935 over the same Web log consisting of 241 Web robots. Comparing these results against those in Table 2, it can be seen that the combined approach is superior to using only a single syntactic parsing technique. This study thus highlights how a traffic pattern analysis approach is stronger than performing only syntactic analysis.

3.2.2 Detection based on resource request patterns

Guo et al. exploit the assumption that robots request only certain resources when they traverse a site (Guo et al. 2005). Hence, they try to detect Web robots based on the patterns in their requested URL resources, giving rise to two new detection algorithms. The first algorithm considers the volume while the second one considers the rate of resource requests.

The first algorithm involves a classification of requested URL resources. The resources are divided into 8 different types, namely Web page, document, script, image, music, video, download, and other. In the first step, entries in the log file are grouped according to the type of resource requested. In the second step, the host and user-agent fields are parsed from each group. For each group, records that carry the same address and user-agent field are treated as a collection of requests coming from a single visitor. If the time interval between two requests from the same visitor is longer than a fixed threshold, then the requests are considered to be from different sessions. All sessions in all groups are formed using this method. In the third step, sessions are marked as a “robot candidate” if a single resource type was requested. In the fourth step, the first three bytes of the IP address and user-agent fields are checked for each “robot candidate” session. Those that match are treated as referring to a single robot. Finally, the number of sessions of the same type generated by the same robot candidate and the number of corresponding visiting records are counted. This data is collected to recognize sessions that only request a single main type of resource, although these may be of significant length. It is possible, however, that humans may also request only one type of resource during their visit, if for example they know the specific location of a resource and navigate to it directly. To address this, the algorithm also considers the number of visits recorded for that client to distinguish robots that make repeated visits against humans that visit the server only once to obtain some resource. If the count of the number of sessions of the same type generated by the same robot candidate and the number of visits recorded both exceed pre-defined thresholds, the session is classified as being from a robot.

The second algorithm considers the rate at which resources are requested as well as the *member list of a Web page*, defined as the aggregation of all URLs of the embedded objects (e.g., images, frames, sounds, etc.) in the page. The idea is to generate a list of the requested resources and the times at which they were requested. If the time difference of all the requests in a list is greater than some threshold, then the visitor should be classified as a robot. Furthermore, if a Web page is accessed but not all embedded objects are requested (e.g. only a subset of a *member list* is requested) then the user can also be suspected to be a robot. The algorithm assumes that the request patterns of humans are governed by the behavior of a common Web browser. Thus, an initial request for an html Web page is followed by a barrage of requests for the embedded resources sent by the browser as it renders the site in real time. Thus, if all the resources are not requested, then the visitor is less likely to be a human.

The detection capabilities of the two algorithms are compared over the same set of data. Their experimental results showed that their algorithms detect all robots that exhibit good behavior (which they define as a robot that requests the file *robots.txt*). In terms of accuracy, of the 253 clients marked as possible robots, 28 clients were positively identified by setting threshold values for the number of sessions of the same type generated by a robot at ≥ 2 and the number of corresponding recorded visits at ≥ 5. They set these thresholds based on the assumption that robots divide their work into several subtasks, and hence, produce more sessions than a human user. Also, they assume that robots request a “bit more” content than a human. Of the 28 clients, 20 were “well behaved” while 8 would have remained undetected as they did not request *robots.txt*.

3.2.3 Detection through query rate patterns

Duskin et al. separate robots from humans in Web search engine logs by analyzing the rate at which search queries are sent (Duskin et al. 2009). They argue that traditional detection techniques are inapplicable to access logs recorded by Web search engines because the logs do not record all the fields in the http request packet that a Web server access log does. Furthermore, the standard approach for detecting robot requests in a search engine access log, based on setting thresholds off of various metrics (Jansen et al. 2000; Buzikashvili 2008), are inadequate. While such a heuristic technique may be useful to filter a majority of Web robots, it is incapable of detecting those that do not send search queries at a rapid rate.

In order to improve upon this heuristic approach, the authors propose that the activity patterns of search users must be studied using multiple metrics, such as the rate of query submission, time interval between queries, rate at which users type, duration of sessions of continuous activity, correlation with time of day, and the regularity of the submitted queries. In this work, the authors examine the behavior using the average rate of queries submitted and the time interval between successive queries. The philosophy of the methodology is to first set thresholds on one attribute in order to partition the sessions into robots and humans, and then subsequently examine the differences in the distribution of the other attribute for the members in the robot and human classes to identify the critical distinctions between the two. Thus, their paper introduces two separate robot detection techniques.

Three data sources are considered: logs from AllTheWeb in 2001 (ATW), AltaVista in 2002 (AV), and MSN in 2006. First, they analyze each log by classifying sessions into Web robots and humans based on the number of queries per session. For the AV and MSN datasets, they classify sessions with less than 10 queries as being a human, and the rest as Web robots. For the ATW dataset, they classify sessions with less than 50 queries as human and sessions with greater than 300 sessions as Web robots. Different thresholds are used for the initial classification because ATW features click-through data, so it is expected that there will be more entries per user relative to the MSN and AV datasets. They also distinguish between different queries and resubmission of the same query many times, which may occur when a user clicks on links from the result page and then returns to the results, or requests additional search results.

Next, the authors classify Web robot and human sessions based on the smallest interval between different queries in a session. They assume that humans cannot submit a new query within 1 s of the previous query, thus sessions containing an interval of less than 1 s are considered to be from robots. Sessions whose smallest time between queries is greater than 25 s in the ATW and MSN datasets and greater than 10 s in the AV dataset are classified as humans. The threshold for the AV dataset was lower because of the shorter delays observed in that log. From this classification, the distributions of the number of queries submitted by Web robots and humans were studied. The results show that the probability of a human exhibiting a given number of queries per session decreases monotonically with the number of queries. It does not monotonically decrease, however, for Web robots.

3.2.4 Detection using traffic metrics

Lin et al. introduce a scheme to categorize user sessions into different groups (Lin et al. 2008). This study postulates that modern Web traffic is multi-class, consisting of humans, Web robots, and other Internet protocols, such as peer-to-peer file sharing. They propose a series of metrics to measure session properties and then use these metrics to heuristically determine whether a session is generated by a human, Web robot, or is P2SP (Peer-to-Server-Peer) traffic.

To develop their proposed metrics, they first assume that "short" sessions (defined as sessions with only a single request) are generated from "gentle crawlers" (a type of Web robot) or P2SP clients, very long sessions are always generated by a Web robot, and that a Web robot session is likely to send requests for a larger variety of resources when compared to human or P2SP clients that target specific information on a Web server. They also assume that Web robots are most interested in html, htm, jsp, and asp files and favor them over all other types of resources.

They define the metric *Human Similarity* (HS) in order to estimate the commonality between sessions generated by humans and other classes of users. The HS of a session is defined as:

$$HS = \sum_{i=1}^{n} s(C_i)w(S_i)$$

where $s(C_i)$ is a score assigned to the response code of the ith request C_i and $w(S_i)$ is the weight of the type of resource requested during the ith request S_i. If S_i corresponds to a file type commonly requested by P2SP and other Web robots, such as mp3, wmv, and exe files, the weight is assigned as $S_i = 10$, otherwise $S_i = 1$. The response code scores are negative for 4xx and 3xx responses, positive for 206 and 200 responses, and 0 for all other codes. No explanation is provided for the assignment of these values. They observe that the probability density function of HS contains four peaks; two that appear near zero on both sides, and two that represent an extremely high and low HS respectively. Thus, they propose three threshold values T_1, T_2, and T_3 to separate the pdf into four regions. For sessions with only one request, if the value of HS falls between threshold values T_1 and T_3, it is classified as a Web robot. Otherwise, the region that the session's HS value falls into is considered along with the additional metrics to classify a session.

The second metric is *Diversity Factor* (DF), which captures how many diverse types of resources are requested per session. Rather than using the number of unique resources requested, the authors use unique resource sizes because modern Web pages deliver content to users through dynamic html, where the same html page may deliver different embedded resources every time the page is requested. To define the metric, they let $R = \{r_i\}$ be the set of all resources requested in a given session. Then, the DF of a session is given by:

$$DF = \frac{|\{s|\exists s \in \mathbb{N}, s \in R\}|}{|\{r_i|r_i \neq 0\}|}$$

Through experimentation, the authors determine that sessions with $DF < 0.5$ are highly suspect to be from Web robots. They also propose an automatic way to find a threshold to classify sessions according to their DF measure, given by the K largest peaks in the pdf of DF across all sessions.

The final metric is *Html Affinity* (HA), which reflects the assumption that human sessions will consist of a mix of resource types while Web robot sessions will be dominated by requests for content files that contain links to resources. HA is given by the percentage of requests in a session that are for an html, htm, jsp, or asp type resource.

These metrics are combined to develop an automatic classifier for Web robot traffic. First, the set of sessions are divided into those that have a single request and more than one request. Sessions whose HS do not fall between T_1 and T_3 are filtered out and considered to be P2SP traffic. For the remaining sessions, those whose DF is below the computed threshold value or whose HA is above 0.65 are considered to be Web robots.

They confirm that their scheme can detect all robots that request the file `robots.txt` from the logs of various Web servers hosted by Tsinghua University. They also confirm that the distribution of the metrics of the categorized traffic are highly consistent across all sessions in a group, even across long time spans.

3.3 Analytical learning techniques

Analytical techniques build on the traffic pattern analysis by developing formal models to represent the characteristics of web robot visits. A properly trained model ensures that a useful robot (i.e., one that does not convey random or arbitrary behavior) will be successfully detected. These analytical models are trained by using robot sessions extracted from the Web logs using a simple syntactic analysis or traffic pattern analysis technique. Analytical techniques can be further classified according to the modeling paradigm and the features of robot traffic considered in detection.

3.3.1 Detection using decision trees

Tan and Kumar attempt to discover Web robot sessions by utilizing a feature vector of the properties of Web sessions (Tan et al. 2002). In the first step, they propose a new approach to extract sessions from log data. They argue that the standard approach based on grouping Web log entries according to their IP address and user-agent fields may not work well since an IP/user-agent pair may contain more than one session (for example, sessions created by Web users that share the same proxy server). Furthermore, a session containing multiple IP addresses or user-agents will become fragmented. Therefore, to determine what session a log entry l belongs to, each active session is scanned to check the time difference between l and the current session, along with some unspecified session contiguity conditions. If this time difference exceeds a threshold or the conditions are not met, then a new session is generated starting with log entry l. Before scanning, all the active sessions are divided into four groups depending on whether the user-agent and IP address fields match those found in l. The first group consists of sessions where both user-agent and IP addresses match, followed by the two session groups with one matching field, and finally the group with no matching fields.

They then derive twenty-five different properties of each session by breaking down the sessions into episodes, where an episode corresponds to a request for an HTML file. Of these, they use only three for the initial classification of sessions. These include checking if *robots.txt* was accessed, the percentage of page requests made with the HEAD http method, and percentage of requests made with an unassigned referrer field. These attributes are used since they most distinctly represent sessions likely to be robots, assuming that normally a human user would not request *robots.txt*, send a large number of http HEAD requests, or send requests with unassigned referrer fields.

From this initial class labeling, the observed user-agent fields are partitioned into groups of known robots, known browsers, possible robots, and possible browsers in the following manner. If a derived session s contains a request for *robots.txt*, the session is declared to be a robot. Otherwise, the user-agent fields of the requests in the session are considered. If s only ever has requests from one user agent, and the user agent is a known robot or a possible robot, then s is labeled as a robot. Otherwise, it is labeled as a human. If s has requests from multiple user agents, however, the session is labeled as a robot only if there are no sessions that are known browsers or possible browsers or if the session contains requests that all use the HEAD http method or requests that all have unassigned referrer fields.

Finally the technique adopts the C4.5 decision tree algorithm over the labeled human and robot sessions using all of the twenty-five derived navigational attributes. Their objective is to develop a good model to predict Web robot sessions based only on access features and to detect robot traffic as early as possible during a robot's visit to the site. This classification model when applied to a data set suggests that robots can be detected with more than 90% accuracy after four requests. They find that the recall (r) and precision (p) of their techniques after more than three requests are greater than 0.82 and 0.95, respectively.

3.3.2 Detection through neural networks

Bomhardt et al. use a neural network to detect Web robots and compare the results to a decision tree technique similar to the one by Tan and Kumar (2005). They develop a Web log pre-processing tool called RDT in order to develop the feature vector for each session.

The RDT log pre-processing tool operates as follows: in the first step, the single log entries are broken into sessions by grouping requests that have matching IP addresses and user-agent fields such that the successive requests are less than 30 min apart. Next, the tool automatically classifies sessions using heuristics known to reliably identify Web robots. If a session contains a request with a login name, or the session IP address is contained in a pre-constructed list of known IP address for human users, the session is labeled as a human. If a session contains a request for a file from a given list of *trapfiles* that may be deployed (resources that should never be requested by a human, for example, typical files used in attacks such as `cmd.exe` or the file `robots.txt`), has a user-agent field that exists in a specified list of known robot agents, or if the session IP address is contained in a list of known robot IP addresses, the session is labeled as a robot.

Using this technique, the RDT is able to reliably generate a set of sessions known to be human or robot agents. Next, the attributes that constitute a feature vector are determined for each robot and human session. Specifically, they use the total number of page requests, percentage of image requests, percentage of html requets, total session time, average time between requests, average standard deviation of time between requests, percentage of error http responses, and percentage of http commands used to retrieve requests. They also include new features such as the total number of bytes sent, total site coverage during the session, and the percentage of responses sent with response codes 200, 2xx, 301, 302, and 304. Many of these attributes overlap with those used by Tan and Kumar (2002). This indicates an emerging consensus on what metrics should be used to characterize Web robot traffic.

Machine learning models of human and Web robot activity are then built using a neural network, logistic regression, and a decision tree. The latter two were used to compare the performance of the neural network. Because several attributes in the feature vectors require at least two requests for their computation, the datasets were partitioned into three groups: *all sessions*, *single-request-sessions*, and sessions with more than two requests.

The authors evaluate their robot detection technique off of two log files, one from an educational website and another from an online shop. For each domain, the data was partitioned so that 40% is used to test the models during construction, 40% is used to train the models, and 20% is used to validate the models. They evaluate these models by measuring their recall, precision, and misclassification rates. The misclassification rates are also compared to a baseline misclassification rate achieved by a trivial model that always labels a session with the same class as sessions in the training data set. For the educational website dataset, they find that the neural network performs best when it is used only for sessions with more than two requests and a decision tree is used for sessions with a single request. The recall ($r = 0.942$) and precision ($p = 0.891$) achieved by this combined approach is comparable to the approach proposed by Tan and Kumar. For the online shop dataset, the neural network offered the best performance after three or more page views were considered in a session, with $r = 0.947$, and $p = 0.954$.

3.3.3 Detection via a Bayesian network

Stassopoulou and Dikaiakos presented a Bayesian approach to crawler detection (Stassopoulou et al. 2007), in which they first identify sessions from the Web log by grouping entries according to their (IP, user-agent field) pairs. Requests are added to a session until the time between a request exceeds a certain timeout. The timeout threshold is dynamic, depending on the number of requests in the session. Then for each session the maximum sustained click-rate, session duration, percentage of image requests, percentage of pdf/ps requests, percentage of responses of 4xx code, and a binary value for if *robots.txt* were accessed are extracted. These features form the nodes (variables) of a naive Bayesian network, where the causal node C represents the session classification (human or robot) and each effected node F_i represents an extracted feature. This framework enables them to combine all pieces of evidence to derive a probability for each hypothesis (i.e., Web robot vs. human).

The network is trained over a data set of thousands of sessions. But rather than manually classifying each session as human or robot for training, they use a heuristic, semi-automatic method. First the sessions are assumed to be human. Then the following heuristics are used to determine if a session must be labeled as a robot: (i) IP addresses of known robots; (ii) the presence of an http request for *robots.txt*; (iii) sessions that last longer than 3 h; and (iv) an HTML-to-image request ratio of more than 10 HTML files per image file. These metrics were then extracted for the heuristically classified sessions and those modeled by continuous distributions such as session duration were quantized into discrete values using the entropy method (Shannon 1948). The initial values for the root node as well as the conditional probability distributions for the non-root nodes were computed from the extracted traffic features. The experiment was performed five separate times, with each trial using different levels of human and robot sessions in the training sets. Their results show a best recall $r = 0.95$ (when the training set contains an equal number of humans and robots) and at its worst a recall of $r = 0.80$ (when the number of human sessions in the training set dominate the number of robot sessions).

3.3.4 Detection by a Hidden Markov model

Lu and Yu use a hidden Markov model (HMM) to distinguish robot and human users based on request arrival patterns (Lu et al. 2006). They argue that a human visitor to a Web page is characterized by a burst of http requests from the browser for all embedded resources, followed by a period of inactivity while the user views the page. A robot, however, sends requests for resources at a slower rate and with a steady period between requests. To capture this phenomenon, the authors partition time into discrete intervals of the same length. One or more requests from the same user that arrive in the same time interval is called a batch arrival. Such a batch arrival is more reflective of a human user than a robot. Each interval with a batch arrival is considered as an observation for the HMM.

The authors use previously observed robot sequences to train their HMM. This training set was obtained by examining the user-agent field and extracting those requests from agents that are clearly robots. Only batch arrivals with more than 30 requests were considered for training, resulting in 612 observed sequences. Future incoming request sequences are fed as input to the trained HMM to compute the likelihood that the request sequence is from a robot. This likelihood is computed by the forward-backward procedure (Rabiner 1989). To test the adequacy of their technique, an equal mix of robot and human sessions from the same logs as the training set were composed to form the test data. For this test data, their HMM yields a detection rate of 0.976 with only 0.02 false positive rate.

3.4 Turing test systems

Turing test systems attempt to classify active sessions in real time by forcing the visitor to take a test in order to determine if the session is human or robot induced, rather than doing a post-mortem analysis of server access logs. The key aspect of these systems is the technique employed to force a user into taking the Turing test.

3.4.1 Detection via CAPTCHA tests

A common example of a Turing test system is the CAPTCHA test, proposed by Ahn et al. (2003). CAPTCHA, which stands for **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part, is a challenge response test embedded in an html page. The server generates a simple test which the user must pass in order to gain access to some resource. Common CAPTCHA tests charge users to copy text from a generated image or type a word or a phrase from a generated sound file. These images and sound files are generated so that computers are unable to recognize the characters from the image or analyze the sound file to determine the word or the phrase spoken. CAPTCHA tests have been employed with great success to prevent robots from gaining access to Web forums, where a robot can easily harvest e-mail and IP addresses of users that post to the forums as well as spam unwanted advertisements on discussion threads. CAPTCHA is also commonly used on pages to create accounts for e-business sites and signing up for online e-mail services.

It is not impossible for a human user to fail a CAPTCHA or to request the server to generate a new test, if the image or the audio distortion is too great to interpret. A human user may also get discouraged if the test is too difficult, and may even terminate the session. Furthermore, it is possible that humans who are only interested in browsing the site for resources accessible without the CAPTCHA test may not take it. Thus, while CAPTCHA can provably classify active sessions that are produced by humans, failing or not taking a CAPTCHA test is not sufficient evidence to classify a session as robot induced.

Building stronger and more sophisticated CAPTCHA systems that are easier for humans to pass, more difficult for Web robots to break through, and applicable in a wider range of domains is an area of active research. This is due to the existence of CAPTCHA-solving services, which Web robots could query in order to bypass a test (Motoyama et al. 2010). Shirali-Shahreza et al. introduce a CAPTCHA designed to counter SMS-Spam messages sent to cell phones (Shirali-Shahreza et al. 2008). When an SMS message is sent, the carrier's SMS routing agent sends a challenge to the sender in the form of an image. The SMS sender then needs to reply with the correct name of the object in the picture before the routing agent forwards the message to the recipient. Gossweiler et al. introduced a new CAPTCHA test that charges users to re-orient an image such that it is facing upright (Gossweiler et al. 2009). This task requires a visual analysis of the patterns in the image in order to perform a geometric modification. Image reorientation can be easily performed by humans compared to CAPTCHAs that use distorted sounds. Such reorientation, however, cannot be performed easily by automated agents. Kluever et al. present a CAPTCHA that challenges the user to label a video and then compare the user input to the video's tags from the online service YouTube (Kluever et al. 2008). Responses are graded based on the string edit distance from the tags available, and through stemming the user responses to improve the likelihood that responses match YouTube tags. They find that the success and failure rate of their video-based CAPTCHA is similar to the traditional CAPTCHA which involves transcribing distorted text, but a majority of participants find the video CAPTCHA to be more enjoyable.

3.4.2 Detection by implicit human browsing behavior

While a Web robot is capable of exhibiting human-like patterns in its traffic, it can still be captured by a turing test system that checks for behavior unique to an interface that humans use to access information on the Web. Park et al. exploit this by developing a technique which requires users to take a Turing test implicitly during a session. In this case, human activity is detected by noting the occurrence of specific events caused by a Web browser (Park et al. 2006). Their technique is also able to differentiate human traffic from robots that request html pages and their embedded resources.

To detect events caused by human activity, for each client request a random key k is generated and paired with the resource requested. This pair is saved in a table indexed by the IP address that the request comes from. A custom Javascript is embedded into the pages served to the client dynamically. The Javascript includes an event handler for mouse movement or key clicks. This event handler will fetch a fake embedded object whose URL contains k. When the event handler sends the request for the fake embedded object, if the k in the URL matches the value of k stored in the server-side table, the session is classified as human. If k does not match, or the fake embedded object is never requested, the session is classified as a robot. The script is obfuscated with additional entries to prevent it from being deciphered by a robot attempting to discover the proper value of k. This test determines if a user is human or robot by determining if mouse or key clicks are ever performed on the Web site.

In practice, some users may disable JavaScript on their browsers. To avoid classifying such users as robots, common patterns of Web browsers are considered as well in a second embedded Turing test. The second technique is based on the observation that many robots crawling specific types of resources, such as only html or only image files, do not download certain objects on a Web page, particularly resources containing html presentation information such as CSS or JavaScript files. To administer this test, a reference to a non-existent CSS file for each html page is dynamically added upon being served. While Web browsers should send a request for this CSS file automatically, robots that do not download presentation related information will not send a request and thus be classified. Similarly, links to 1-pixel transparent images and silent audio files are also placed as transparent resources in the html. Because the link is invisible humans should not request these resources, while robots may analyze the html and make requests.

This technique is implemented in the CoDeeN content distribution network (Park et al. 2006). The results show that 95% of human users are detected within the first 57 requests made, with a false positive rate of 2.4%. These results confirm that using both techniques together accurately separates robots requesting html and embedded resources from human visitors. Robots requesting non-html documents exclusively during a session are not detected, however, as only clients that request html documents become subject to the implicit Turing test.

3.4.3 Detection of DDoS botnet attacks

Kandula et al. presented a system implemented as an extension to the Linux kernel, called Kill-Bots, that is designed to protect Web servers from DDoS attacks posing as a flash crowd event (Kandula et al. 2005). A flash crowd event is a phenomenon where the server experiences a sudden surge in the number of requests it receives. Such DDoS attacks may be launched by *zombies*, a type of Web robot installed on infected computers and controlled by a single mastermind. This mastermind usually has direct control of thousands or hundreds of thousands of zombie robots which are all directed to execute the same command. In this case, the zombie robots to be detected are those used to execute a DDoS attack on a Web server.

Kill-bots is a two function system, combining user authentication with admission control. The two-stage authentication mechanism is activated when the server becomes overloaded with active sessions. First, a CAPTCHA test is administered before any site resources can be accessed. While legitimate users will solve the test or at least try to solve the test several times before getting frustrated and leave the site, zombies will continue to send new requests without attempting to solve the test. As a result, if the number of unsolved puzzles from a session exceeds a certain threshold, the IP address of the session is logged as a zombie and future requests from that IP address are discarded. The second stage of authentication is reached when the size of the set of zombie IP addresses stabilizes. In this stage, the CAPTCHA test is no longer administered and the Bloom filter is relied on exclusively to determine if incoming requests from an IP address should be dropped. This stage enables legitimate users who failed the CAPTCHA test and left in discouragement to return and use the site even during a DDoS attack.

If too many server resources are consumed by the authentication process, the remaining available resources may not be sufficient to provide a reasonable quality of service to the human users who are successfully authenticated into the system. To address this problem, the second function of Kill-bots is an admission control system. The admission control will serve puzzles to new, unauthenticated users with an admission probability α^*, calculated as a function of the current arrival rate of attacking requests, legitimate requests, average time to serve puzzles and requests, average number of requests per legitimate session, fraction of the time the server spends in authentication and serving, and the idle time. A request from an IP address which has not yet been authenticated with the server but not blocked will be granted access with probability α^* and dropped with probability $1-\alpha^*$. Being adaptive, α^* self adjusts as the values of the parameters change during an attack so that server goodput is always maximized.

The system was evaluated by launching a botnet attack on a local Web server using the PlanetLab wide area network (PlanetLab 2003). Legitimate clients were also setup on the same LAN as the Web server. The results show that their system is able to quickly adapt during the first stage of the authentication process, causing the mean response time of requests to expeditiously drop to a value close to the one when there is no attack. When the second part of the authentication process is activated, a large increase in goodput is observed due to reduced detection cost and users accessing the system that previously could not pass the CAPTCHA test. Despite the ongoing DDoS attack, the server performance is close to the performance when it is not under attack. Thus, the Kill-Bot system can properly drop all requests from zombies and produce a correct list of all sessions induced by them.

4 Limitations in the state-of-the-art

The techniques presented in this survey reflect major milestones in Web robot detection over the past decade and also demonstrate how widely varied detection methods can lead to an effective solution. These techniques have evolved from simple access log parsing, to using analytical models, to solutions engineered for specific types of robots and traffic. In this section, we first compare the existing classes of techniques to identify which philosophy may be most effective at detecting both common and unknown, evasive robots. Subsequently, we identify the issues involved in the implementation of this detection philosophy that we conjecture to be the most effective.

4.1 Comparing existing techniques

It is difficult for a third party to experimentally compare the results of these published detection techniques for two reasons. First, each detection technique relies on specialized tools or operating system modifications, which would need to be implemented faithfully. For sophisticated system-level modifications, such as those used in Kill-bots, a third party implementation cannot be guaranteed to have high fidelity. Second, it is expected that the accuracy of some techniques will depend on the data set used for demonstration. For example, the technique based on navigational patterns may exhibit poor accuracy if the data is from a Web site with very few pages, making the navigational pattern of a robot similar to most humans because they are likely to cover the majority of a site during a session. Furthermore, because the sessions are so short on these types of Web sites, there may not be enough observations for learning techniques to reach valid conclusions. Thus, at least a syntactic analysis approach may be the most effective type on such Web sites. As another example, consider an access log from a very large Web site that contains entries from a variety of robots with different navigational patterns. In this case, the navigational pattern and learning techniques will naturally be more effective than syntactic log analysis. These examples illustrate that the prevalent techniques will need to be compared using a variety of logs with varying features and levels of robot activity to obtain any meaningful conclusions. These data-intensive demands, the need for proper operating environments and hardware for each technique, and the lack of implementation details to faithfully reproduce each detection strategy make a third party comparison of the existing techniques unrealistic.

The works presented can thus be best compared by considering the strengths and weaknesses of their underlying detection philosophy. Syntactic log analysis relies only on a textual analysis of the Web logs that considers only explicit recorded information. Traffic analysis techniques improve upon syntactic analysis by considering important facets about a robot's traffic that may be implicit in the logs. Analytical learning models are considered to be even stronger than traffic analysis as they are able to consider the relationships among these implicit facets that are considered separately in traffic pattern analysis. Turing test systems may be considered to be strongest, as they may be able to classify all robots of a specific type on the Web server in real time. Because they are engineered to detect only specific types of robots (such as those used in DDoS attacks or those that request at least one html document), however, their power is limited.

Analytical learning techniques may also be superior in the detection of evasive robots compared to the other three types of techniques (syntactic log analysis, traffic pattern analysis and Turing test systems). This is because the other three techniques rely on some procedure, algorithm, or system which a savvy author can engineer a robot to evade. The analytical detection techniques, however, do not incur this disadvantage because they rely on learning the features of robot traffic on the server. As a result, it may be difficult for robot authors to circumvent detection by the analytical techniques. In fact, the only way to avoid detection by an analytical learning technique is to have a robot emulate the visiting patterns of a human, something that is not simple to achieve. For example, human users frequently traverse back and forth between pages, reference information, and/or navigate back to a main page to click on a different link. Such back-and-forth navigation cannot be accurately emulated by robots, unless a robot was programmed to follow a specific sequence of requests and wait a specific duration between requests that mimic a human visitor. Furthermore, a robot that crawls like a human would discover only a small amount of information, which may or may not be meaningful. Moreover, robots that mimic humans will not impose any undue strain on the Web server. Hence, we should be comfortable allowing such robot traffic to go undetected.

In summary, our discussion suggests that analytical learning models are the most promising types of robot detection techniques because they not only consider implicit characteristics of robot behavior but also try to explore the relationship between these implicit characteristics using formal models. There remain several issues in the implementation of such analytical learning techniques, however, which we discuss next.

4.2 Implementation issues

Although analytical models are philosophically the most suited for detecting Web robots, there are two key implementation issues that prevent these techniques from being effective outside of their experimental evaluation. The first issue is that analytical models need training data which faithfully represents modern Web robot traffic. The second issue is that they rely on the statistical differences between human and robot traffic, and these differences may change as the robot traffic evolves over time. We examine these implementation issues in detail next.

4.2.1 Representative training data

Analytical learning models need to be trained over a known set of data. Thus, the accuracy of their results will depend on how faithfully the training sample reflects robot behavior. For example, the training data used in prior studies was extracted manually by a human expert or by using a weak detection method that has high precision, such as checking for sessions that request `robots.txt`. Such training data, however, is unlikely to consider all Web robots, including those that evade detection by providing fake values in their user-agent fields, use proxy servers to mask their IP address, or ignore `robots.txt`. To improve the detection accuracy of analytical models, it is thus necessary to determine the percentage of traffic due to hidden or evasive robots, and then add a proportional amount of such traffic to the training set.

Strong robot detection techniques are required, however, to detect such evasive and hidden robots. This leads to a dilemma: being able to use a stronger, more flexible detection technique (analytical rather than algorithmic) requires a training set reflective of robot traffic on the server, something that can only be achieved by detecting hidden traffic.

4.2.2 Evolving traffic characteristics

Another weakness in the existing implementations of analytical learning models is that they use the statistical properties of Web robot traffic in the development and training of the model. It is likely that robots behave differently depending on the domain of the server and the type of services it provides. For example, a price comparison or a shop bot crawling a server in an academic domain is unlikely to send many requests as it realizes that there is no capability to search for goods. Similarly, an indexing robot will end its crawls prematurely when faced with a site that requires some authentication to access, while another robot with specialized functionality may have the credentials needed to enter a site and perform a complete crawl.

Furthermore, there is evidence to suggest that robot traffic on the Web is significantly evolving. This evolution is natural considering how rapidly new value-added services are being developed, which require support from more advanced Web robots to fetch information. For example, state-of-the-art search engines provide users with additional complex results for a query, such as related images and video, content previews of links in the search response, results from related queries, and links to the best prices of goods related to the query if appropriate. Another reason for this evolution is the rise of *Web 2.0* services that encourage users to upload new information to share with the world dynamically. Web blogs, social networking sites, user comments on news sites, and media uploading services such as YouTube have caused a surge in the amount of dynamic, time-sensitive information posted to the web. This has forced Web robots to make more frequent, demanding crawls with sophisticated techniques in order to stay abreast with the latest available information. Thus, an analytical detection scheme that relies on a snapshot of robot traffic at any given point of time is sure to become less effective over time.

5 Proposed detection approach

In this section, we discuss how these implementation issues may be addressed to enable strong robot detection in practice. To address the problem of incomplete training data sets, we propose a detailed study to classify Web robots and also suggest a suitable detection strategy for each class. To address the problem of evolving statistical traffic patterns, we suggest that the analytical models instead capture fundamental distinctions between Web robot and human traffic, which lie within their navigational and resource request patterns.

5.1 Training sets

We propose to partition all robots into a collection of classes, such as indexer, harvester, or unknown robots, and then study the fundamental properties of the traffic from each class. Through this study, we expect to identify the behavioral characteristics of robots from each class, which will suggest the type of technique that is effective for each class. Previously published work regarding trends about robot traffic (Smith et al. 2006; Dikaiakos et al. 2005; Doran et al. 2008; Ye et al. 2004; Doran et al. 2009; Lee et al. 2009) may also guide in classifying robots and determining their intrinsic properties. For example, if the robots from a given class exhibit similarities in their navigational patterns, a traffic analysis technique could be used to extract them from the logs. As another example, if some class is comprised entirely of robots with well-known user-agent fields, a syntactic analysis technique would be best.

The greatest challenge lies in building a classifier for hidden robots because their traffic cannot be easily identified in Web logs. For such robots, detection based on traffic patterns or Turing test systems may be employed so that the aspects of the hidden robot behavior can be recorded and then assessed offline. The selection of a Turing test system should consider the feasibility of its implementation, since such detection systems can be very difficult to replicate. The training set can be composed from the robot sessions detected using each technique. Because the analytical model will now be trained using a proper balance of all types of robot traffic, it will be able to detect most robot traffic, leading to a generally applicable solution to the robot detection problem.

The above discussion suggests that all the other types of detection approaches must be used to build complete training data sets for an analytical model. A legitimate question then arises as to why these techniques cannot be used in tandem to detect Web robots. There are two critical disadvantages in taking this approach. First, neither one of the above three techniques is sufficient to detect all types of robots. Thus, it will always be necessary to use all three types of techniques simultaneously. Running this collection of techniques would be very expensive both computationally and in terms of time since it would require a series of different analysis on many access logs. By contrast, an analytical learning technique only needs to run a single analysis over the Web logs. If this analysis is done frequently, such that the set of new entries in the logs is small, it will be relatively inexpensive. Frequent examination of a small volume of traffic also enables the analytical technique to continuously generate an updated picture of the robot traffic found on the Web server. Finally, as discussed in Sect. 4.1, a savvy robot author may be able to evade detection by the three techniques through careful engineering, but may not be able to fool the analytical learning models.

5.2 Fundamental traffic distinctions

We argue that analytical techniques must be based on fundamental distinctions between Web robot and human traffic that are expected to remain invariant over time in order to keep techniques effective across server domains and in the face of evolving robot traffic. We believe that such distinctions lie within the navigational and resource request patterns of robots and human clients.

The navigational patterns of humans represent the action of following a series of links on web pages in order to find information, restricted by the link structure of a site. Human patterns may also include frequent back-and-forth navigation through a site, using a Web browser's history, "back", and "forward" feature. Loops may also be present if a human becomes disoriented during their visit. In contrast, robots are neither expected to have such complex navigational patterns, nor would they be restricted by the link structure of the Web site. After an initial crawl of a site, robots are capable of learning precisely where the information that they are seeking resides, so that on repeated visits they may only send requests for specific files or restrict their crawling to specific areas of the site. Furthermore, well-behaved robots may volunteer to restrict their crawls only to those areas of the site identified by the server administrator, as specified by the file `robots.txt`. Therefore, because humans will always need to rely on the structure of a site to find information, and because robots will never need to restrict their navigation due to the site structure, the navigational patterns exhibited by humans and robots will always be fundamentally different.

Properties of the resources requested per session is another distinction between robot and human traffic that is not expected to change over time. This distinction arises because human users retrieve information off of the Web via some interface, such as a Web browser. This interface forces the user's session to request additional resources automatically. Most Web browsers, for example, retrieve the html page requested, parse through it, and then send a barrage of requests to the server for embedded resources on the page such as images, streaming videos, and client side scripts to execute. These scripts may also cause additional artifacts on the page to be requested and then displayed to the user, such as pop-up windows or advertisements within the Web page being viewed. Thus, the temporal resource request patterns of human visitors are best represented as short bursts of a large volume of requests followed by a period of little activity. In contrast, Web robots are able to make their own decisions about what resources linked on an html to request and may choose to execute the scripts available on a site if they have the capacity to do so.

6 Concluding remarks

In this paper, we proposed a framework to classify contemporary Web robot detection techniques. The objective of the framework is to exploit the commonalities across the prevalent techniques, while drawing attention to their differences. Under this framework, we defined four classes to partition the existing techniques, depending on the philosophy of the underlying detection strategy. Through a critical analysis and comparison of the four categories, we argue that analytical learning techniques have the most potential to provide robust Web robot detection. The current implementations of the analytical learning techniques, however, suffer from two key limitations. To address these limitations, we propose two new strategies. The first approach proposes a composite technique to develop training sets that more faithfully represent modern Web robot traffic. The second approach bases the analytical learning technique on fundamental distinctions between Web robot and human behavior rather than in the statistical properties of their traffic.

Our future research comprises of developing appropriate classifications as well as conducting an in-depth analysis of traffic from a variety of robots with varying demands and functionality to gain insights towards the development of training sets for strong robot detection systems. In addition, a new HMM-based Web robot detection technique that incorporates our approaches to improve Web robot detection will be developed and analyzed.

References

Ah LV, Blum M, Langford J (2003) CAPTCHA: using hard AI problems for security. In: Proceedings of Eurocrypt, pp 294–311

AWStats—Free log file analyzer for advanced statistics (GNU GPL). Available at http://awstats.sourceforge.net/

Bomhardt C, Gaul W, Schmidt-Thieme L (2005) Web robot detection—preprocessing web logfiles for robot detection. In: New developments in classification and data analysis, pp 113–124

Buzikashvili N (2008) Query log analysis: disrupted query chains and adaptive segmentation. In: Proceedings of workshop information. Retrieval 2008, pp 35–40

Dikaiakos MD, Stassopoulou A, Papageorgiou L (2005) An investigation of Web crawler behavior: characterization and metrics. Comput Commun 28:880–897

Doran D, Gokhale SS (2009) Classifying Web robots by K-means clustering. In: Proceedings of the international conference on software engineering and knowledge engineering, pp 97–102

Doran D, Gokhale SS (2008) Discovering new trends in Web robot traffic through functional classification. In: Proceedings of international symposium on network computing and applications, pp 275–278

Duskin O, Feitelson DG (2009) Distinguishing humans from robots in web search logs: preliminary results using query rates and intervals. In: Proceedings of 2009 workshop on Web Search Click Data, pp 15–19

Geens N, Juysmans J, Vanthienen J (2006) Evaluation of Web robot discovery techniques: a benchmarking study. In: Lecture notes in computer science vol 4065/2006, pp 121–130

Giles C, Sun Y, Councill I (2010) Measuring the web crawler ethics. In: Proceedings of 19th international conference on the World Wide Web, pp 1101–1102

Gossweiler R, Kamvar M, Baluja S (2009) What's up CAPTCHA?: a CAPTCHA based on image orientation. In: Proceedings of 18th international conference on World wide web, pp 841–850

Guo W, Ju S, Gu Y (2005) Web robot detection techniques based on statistics of their requested URL resources. In: Proceedings of ninth international conference on computer supported cooperative work in design, pp 302–306

Huntington P, Nicholas D, Jamali HR (2008) Web robot detection in the scholarly information environment. J Info Sci 34:726–741

Jansen BJ, Spink A, Saracevic T (2000) Real life, real users, and real needs: a study and analysis of user queries on the web. Info Process Manage 36:207–227

Kabe T, Miyazaki M (2000) Determining WWW user-agents from server access log. In: Proceedings of seventh international conference on parallel and distributed systems, pp 173–178

Kandula S, Katabi D, Jacob M, Berger A (2005) Botz-4-sale: surviving organized DDoS attacks that mimic flash crowds. In: Proceedings of the 2nd conference on symposium on networked systems design & implementation, pp 287–300

Kluever KA, Zanibbi R (2008) Video CAPTCHAs: usability vs. security. In: Proceedings of IEEE Western New York Image Processing Workshop 2008

Koster M (1994) A standard for robot exclusion. http://www.robotstxt.org/wc/exclusion.html

Lee J, Cha S, Lee S, Lee H (2009) Classification of web robots: an empirical study based on over one billion requests. Comput Secur 28:795–802

Lin X, Quan L, Wu H (2008) An automatic scheme to categorize user sessions in modern HTTP traffic. In: Proceedings of IEEE global telecommunications conference 2008, pp 1–6

Lu WZ, Yu SZ (2006) Web robot detection based on hidden Markov model. In: Proceedings of international conference on communications, circuits and systems, pp 1806–1810

Motoyama M, Levchenko K, Kanich C, McCoy D, Voelker G, Savage S (2010) CAPTCHAs—understanding CAPTCHA solving from an economic context. In: Proceedings of the USENIX security symposium 2010

Oriley T (2007) What is Web 2.0: Design patterns and business models for the next generation of software. In: Communications & Strategies, pp 17–37

Park KS, Pai V, Lee KW, Calo S (2006) Securing Web service by automatic robot detection. In: Proceedings of the annual conference on USENIX '06 annual technical conference

Princeton University (2003) PlanetLab—an open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org

Prince MB, Holloway L, Keller AM (2005) Understanding how spammers steam your e-mail address: An analysis of the first six months of data from project honey pot. In: Second conference on Email and Anti-Spam

Rabiner LR (1990) A tutorial on hidden markov models and selected applications in speech recognition. In: Proceedings of the IEEE, 77:257–286

Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27:379–423, 623–656

Shirali-Shahreza HM, Shirali-ShahrezaM (2008) An Anti-SMS-Spam using CAPTCHA. In: Proceedings of 2008 ISECS international colloquium on computing, communication, control, and management, pp 318–321

Smith JA, McCown F, Nelson ML (2006) Observed Web robot behavior on decaying Web subsites. In: D-Lib Magazine vol 12. http://www.dlib.org/dlib/february06/smith/02smith.html

Stassopoulou A, Dikaiakos MD 2007 A probabilistic reasoning approach for discovering Web crawler sessions. In: APWeb/WAIM, pp 265–272

Tan PN, Kumar V (2002) Discovery of Web robot sessions based on their navigational patterns. Data Min Knowl Discov 6(1):9–35

Turing A (1950) Computing machinery and intelligence. Mind 59:433–460

Ye S, Lu G, Li X (2004) Workload-aware Web crawling and server workload detection. In: Proceedings of the second Asia-Pacific advanced network research workshop, pp 263–269