Detecting Web Robots Using Passive Request Headers

Farhan Jiva
University of Georgia
jiva@cs.uga.edu

Abstract

Web robots are computer programs that autonomously traverse the World Wide Web via its hyperlink structure. The purposes of these web robots range from beneficial, such as performing indexing operations for search engines, to malicious, such as harvesting email addresses for spam purposes, performing distributed denial-of-service attacks, casting automatic votes, and performing click-fraud. Several reasons exist for the need to identify web robots and distinguish them from legitimate users. Firstly, many e-commerce websites can be adversely affected by unauthorized web robots which gather business intelligence from their website. Secondly, web robots often consume a considerable amount network bandwidth which can affect legitimate users. Finally, it is rather difficult to perform clickstream analysis effectively on sessions created by web robots. Conventional methods for detecting web robots often include relying on IP address and user-agent string information contained with an HTTP request header to perform identification. While these methods can be applicable to some extent, they may not be sufficient to detect those web robots which are stealthy (i.e. arriving from a new IP address and explicitly forging the user-agent string).

In this paper, we perform classification of known web robots against web browsers using a subset of passive HTTP request headers and achieve a classification accuracy of 99% with very low false positives and false negatives consistently over a variety of classification techniques.

1 Introduction

Web robots are automated scripts or agents which perform a specific task without the need for a continuous human operator. Because of the excessive growth of the World Wide Web, web robots have become indispensable tools for performing a variety tasks, such as crawling web sites for indexing search engines to the malicious variety, which include harvesting intelligence to performing other malicious acts. Unfortunately, many of these web robots can cause some level of hindrance to websites. Many websites operate with restrictions on the amount of network bandwidth it is allowed to consume in a certain time frame. Violations to this restriction often cause websites to either go off-line or incur additional inflated costs as a penalty. Because many web robots are highly automated, they can consume a vast quantity of this limited network bandwidth in a short amount of time. This causes severe interference for the legitimate users of the website. Additionally, many web robots can adversely affect a website's ability to perform clickstream analysis. Clickstream analysis is the recording of what a user clicks on while visiting a web site which is then used for web activity analysis, software testing, and market research. Because sessions related to web robot activity are autonomously generated by a computer program, it contributes a significant amount of noise with respect to this type of analysis. As a final example, many web robots are deployed by business organizations to collect email addresses and online resumes, monitor product prices of competitors, and retrieve other information related to competing business organizations.

In many of the situations described, it is often desirable to identify web robots and differentiate them from legitimate users. Current heuristic approaches include examining web server log files for IP address information and the user-agent string[1]. In this paper, we use an approach which does not rely on IP address or user-agent string information. Instead, it involves classification of known web bots against web browsers using a subset of information contained within the HTTP request header.

The rest of the paper is laid out as follows:

- In Section 2 we discuss some related work.
- In Section 3 we discuss some background information.
- Section 4 covers our evaluation, along with our experimental setup and experimental results.
- In Section 5, we discuss our conclusions.

2 Related Work

Several methods exist in the realm of web robot detection. In [3], *Tan et al.* use behavioral patterns of web robots and humans to build classification models with features based on time-related information for each session. In their study, they were able classify web robots with 90% accuracy after examining at least 4 requests within any given session.

In [1], *Park et al.* use a more active approach of identifying web robots versus web browsers by interrogating the web-client for tell-tale signs. They use two main methods in their approach. The first method is to detect keyboard and mouse activity in the web-client and classify the web-client as a web browser if such activity is detected. The second method is to embed invisible links in the web-page with the intuition that human users should not click on them because they cannot see it, opposed to a web robot which follows every link blindly. Using this approach, they were able to detect 95% of humans after examining at least 57 requests with less than a 2.4% false positive rate.

Finally, in [2], *Stassopoulou et al.* use a probabilistic reasoning approach for overcoming the web robot detection problem by using Bayesian-network classifiers. They use a number of features including number of clicks in a given time frame, timing information between clicks, whether the web-client accessed images or other non-HTML resources, access to `robots.txt`, and number of error responses to achieve an 85% to 91.4% classification accuracy amongst a variety of datasets with precision and recall consistently reaching above 95%.

3 Background

The Hypertext Transfer Protocol (HTTP) is an application-level networking protocol which is the basis for data communication for the World Wide Web. This communication takes place between two nodes, the web-server and the web-client. The protocol itself is presented in the form of an HTTP response/request between the web-server and the web-client respectively. Whenever a web-client needs to access a resource on a web-server, it issues an HTTP request to the web-server. Contained within this HTTP request is a variety of attribute-value pairings which provide the web-server with information such as which HTTP protocol version it is using, which media types the web-client can accept, which languages and encodings the web-client is able to use, information about how long to keep the connection open, the user-agent string, amongst others.

Figures 1 and 2 display examples of passive request headers of web browsers and web robots, respectively.

Figure 1: Example of a passive request header from a web browser

```
GET /index.php HTTP/1.1
Host:  x.tow.ly
User-Agent:  Apple Mac OS X v10.6.5
CoreMedia v1.0.0.10H574
Accept:  */*
Accept-Language:  en-us
Accept-Encoding:  gzip, deflate
Connection:  keep-alive
```

Figure 2: Example of a passive request header from a web robot

```
GET /robots.txt HTTP/1.1
Cache-Control:  no-cache
Connection:  close
Pragma:  no-cache
Accept:  */*
Accept-Encoding:  deflate
From:  bingbot(at)microsoft.com
Host:  columbia.cs.uga.edu
User-Agent:Mozilla/5.0 (compatible;
bingbot/2.0; +http
```

We use the term *passive* to describe the method by which this HTTP request is obtained. Passive methods of data collection means that no querying was done to the web-client to retrieve this HTTP request information. This is opposed to *active* methods of data collection, which require some amount of interrogation to the web-client to retrieve information.

- **Web browser labeling:** We set a unique cookie value for each web-clients which accessed our data-collecting web-server. For each returning web-client, we check to see if it has any cookies stored. If it does, and if the unique cookie value matches one that was previously set by us, we label the instance as a web browser. The intuition is that since most web robots do not store cookie information and web browsers will store cookie information, the web-clients seen with a known cookie value are web browsers.

4 Evaluation

In the following section, we will discuss our dataset as well as the types of classifiers used and classification parameters used.

4.1 Dataset

Our dataset consists of 257,114 passive HTTP request header instances, with 190,697 instances being from web browsers and 66,417 instances being from known web robots. The collection process of this dataset spans the length of over one year and was obtained from the web-server running the free image-hosting service `http://tow.ly`.

Dataset labeling. We took certain steps for labeling our data.

- **Web robot labeling:** We used the web robot database located at `http://www.robotstxt.org/`. Using this database, we obtained AS[2] information related to each IP address and labeled those instances whose IP address fell within the range of known IP addresses that performed web bot activities.

Preprocessing the dataset. Recall that an HTTP request header contains attribute-value pairings with information about the web-client. Some of these attribute-value pairings contain values that, if used during classification, would result in misclassification or would receive a high value for information gain. Some examples of attribute-values which were removed are: pairings which contained IP address information, date information, cookie information, referrer information, hostname information, and user-agent string information. The main reason for removing the user-agent information is because it is easily forged, and using its value during classification may cause misclassification.

Initial selection of attributes. Because it is the case that some attribute-value pairings exist for only a small subset of web-clients, we initially choose only those attributes which have values for at least 100 instances. Out of the 145 total attributes in the dataset, 18 of these attributes were selected based on frequency. These attributes are described in the following table.

Header attribute	Description
if-none-match	Allows a 304 Not Modified to be returned if content is unchanged.
version	The version of an HTTP message.
accept	Content-Types that are acceptable.
accept-language	Acceptable languages for response.
accept-encoding	Acceptable encodings.
accept-charset	Character sets that are acceptable.
keep-alive	Contains timeout information about persistent connections.
connection	Allows the sender to specify options that are desired for that particular connection.
content-type	The mime type of the body of the request.
pragma	Implementation-specific headers that may have various effects anywhere along the request-response chain.
cache-control	Used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain.
x-bluecoat-via	Client is behind a BlueCoat proxy.
via	Informs the server of proxies through which the request was sent.
x-wap-profile	A reference to the user-agent profile.
ua-cpu	CPU architecture.
expect	Used to indicate that particular server behaviors are required by the client.
te	The form of encoding used to safely transfer the entity to the client.
content-transfer-encoding	The form of encoding used to safely transfer the entity to the client.

Attribute-value representation. Each of the attribute-value pairings were converted to lowercase (to avoid possible misclassification). Furthermore, each possible value over the set of attributes was then mapped to an integer value.[3] We assigned a special `null` value (mapped to its own integer value) for those attributes which had missing attributes.

4.2 Classification methods and parameters

Four types of supervised learning methods were chosen to perform classification for this study. In the following section we describe the different classifiers and the parameters under which they operate. For all of the experiments performed, we use the `Weka` software library.[4] Given the relatively large size of the dataset we used, we dedicate 66% of the entire dataset for training purposes, and the remaining 34% for testing.

Decision tree. Decision tree learning is a method which is commonly used in machine learning and data mining. Its goal is to create a model that predicts the value of a target variable based on several input variables. For this study, we use the J48 classifier in `Weka`. We use a confidence factor of 0.25 used for pruning, and at least 2 instances per leaf.

Naive Bayes. A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong independence assumptions.

k-nearest neighbor. The k-nearest neighbor algorithm is a method for classifying objects based on closest training examples in the feature space. For this study, we use one-nearest neighbor which uses a linear nearest neighbor search algorithm based on the Euclidean distance metric.

Support vector machines. Support Vector Machines perform classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories. For this study, we use the `LibSVM`[5] plug-in for `Weka`. We use a linear kernel type in the interest of time.

Stacking. Stacking is a type of ensemble learning technique in which multiple classifiers are combined with the intuition that together they will provide better results than only a single type of classifier. For this study, we combine 3 different classifiers, (decision trees, naive Bayes, and k-nearest neighbors) and use a decision tree as the meta-learner. We decide to leave out support vector machines in the interest of time.

Feature selection. Feature selection is the technique of selecting a subset of relevant features for building robust learning models. In this study, we apply a feature selection technique on the 18 total attributes and then perform classification on a subset of attributes to observe any increase in accuracy.

4.3 Experimental Results

We present our classification results in the following section.

4.3.1 Initial classification results

In this subsection, we present the classification results obtained from using the 18 pre-selected attributes over each of the classifiers.

Decision tree. The following table and subsequent confusion matrix describes our classification results using a decision tree.

Correctly classified	**Incorrectly classified**	**Mean absolute error**	**Precision**	**Recall**
99.9828%	0.0172%	0.0003	1	1

a	**b**	**← classified as**
22623	6	a = 1
9	64781	b = 0

Using decision tree learning, we were able to reach a 99.98% classification accuracy. As we can see from the confusion matrix, we were also able to achieve a very low number for false positives and false negatives.

Naive Bayes. We describe our results of the naive Bayes classifier in the following table and confusion matrix.

Correctly classified	**Incorrectly classified**	**Mean absolute error**	**Precision**	**Recall**
99.8845%	0.1155%	0.0012	0.999	0.999

a	**b**	**← classified as**
22629	0	a = 1
101	64689	b = 0

Using a naive Bayes classifier, we were able to achieve a 99.88% classification accuracy with a very low mean absolute error of 0.0012. Compared with our decision tree classifier, the naive Bayes classifier produced a slightly higher number of false negatives, however with the number of false positives being lower.

k-nearest neighbor. In the following table and confusion matrix, we show our results for the k-nearest neighbor classifier.

Correctly classified	**Incorrectly classified**	**Mean absolute error**	**Precision**	**Recall**
99.9806%	0.0194%	0.0003	1	1

a	**b**	**← classified as**
22623	6	a = 1
11	64779	b = 0

Similar to the decision tree learner, we were able to exceed a 99% classification accuracy with very low false positives and false negatives using a k-nearest neighbor approach.

Support vector machines. We describe the classification results and related confusion matrix for our support vector machine classifier.

Correctly classified	**Incorrectly classified**	**Mean absolute error**	**Precision**	**Recall**
99.9554%	0.0446%	0.0004	1	1

a	**b**	**← classified as**
22629	0	a = 1
39	64751	b = 0

Similar to the other classification techniques, we received a very high value for classification accuracy at 99.9554%. Although the classification accuracy was nearly that of what was received for decision trees, we can see that the number of false negatives was slightly higher.

4.3.2 Ensemble classification results

In this next subsection, we present our results obtained from the ensemble learning technique, stacking. Recall that we combine 3 different classifiers, (decision trees, naive Bayes, and k-nearest neighbors) and use a decision tree as the meta-learner. We decided to leave out support vector machines in the interest of time. The following includes results for classification and the related confusion matrix.

Correctly classified	Incorrectly classified	Mean absolute error	Precision	Recall
99.9142%	0.0858%	0.001	0.999	0.999

a	b	← classified as
22564	65	a = 1
10	64780	b = 0

As with our prior results, we see a very high value for classification accuracy at 99.9142% with very low numbers of false positives and false negatives. We find that in our domain, stacking did not produce a significantly higher accuracy, and is comparably to using any of the classifiers independently.

4.3.3 Feature selection classification results

In our final round of testing, we present the effect that feature selection had on classification accuracy.

Selected features. For our attribute evaluator, we use a method which evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. We use a best first method which searches the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility. The results of feature selection show that the attributes `connection` and `content-type` are the most effective attributes to examine in order to make a successful classification. We will now describe our classification results obtained by using these two attributes over the classifiers.

Decision tree. The following table and confusion matrix show our results of classification using a decision tree over the attributes selected from feature selection.

Correctly classified	Incorrectly classified	Mean absolute error	Precision	Recall
99.897%	0.103%	0.0021	0.999	0.999

a	b	← classified as
22629	0	a = 1
90	64700	b = 0

We observe that by using these two selected attributes, classification accuracy remains over 99%. We also observe that the number of false positives has dropped to 0, with the number of false negatives increasing only slightly.

Naive Bayes. We display the results obtained from the naive Bayes classifier using the selected attributes.

Correctly classified	Incorrectly classified	Mean absolute error	Precision	Recall
99.897%	0.103%	0.0012	0.999	0.999

a	b	← classified as
22629	0	a = 1
90	64700	b = 0

We observe that our results are nearly identical to using the decision tree classifier, with the only change being to the mean absolute error.

k-nearest neighbor. The following table and confusion matrix describe the results obtained from a k-nearest neighbor approach using the selected attributes.

Correctly classified	Incorrectly classified	Mean absolute error	Precision	Recall
99.897%	0.103%	0.002	0.999	0.999

a	b	← classified as
22629	0	a = 1
90	64700	b = 0

We note once again that the results from using a k-nearest neighbor classifier is almost exactly the same as using the decision tree or naive Bayes classifier, with the only difference being the the mean absolute error.

Support vector machines. Finally, we describe our results for the selected attributes on our support vector machine classifier.

Correctly classified	**Incorrectly classified**	**Mean absolute error**	**Precision**	**Recall**
99.8948%	0.1052%	0.0011	0.999	0.999

a	**b**	**← classified as**
22627	2	a = 1
90	64700	b = 0

We observe that unlike the previous three classifiers, the results for support vector machine are different but insignificant. The resulting classification accuracy remains high with the number of false positives and false negatives remaining relatively low.

5 Conclusions

Throughout the course of this study, we find that we can detect and classify web robots against web browsers with above 99% accuracy using information from an HTTP request header. We find that no matter which classification technique is used and whether or not ensemble learning is performed, we receive an exceedingly high classification accuracy with a very low number of false positives and false negatives. Furthermore, we learned that by performing feature selection over the set of attributes, that the attributes `connection` and `content-type` are the most effective attributes to examine in order to make a successful classification.

References

[1] K. Park, V. S. Pai, K.-W. Lee, and S. Calo. *Securing Web service by automatic robot detection*. In USENIX Technical Conference, June 2006.

[2] Athena Stassopoulou , Marios D. Dikaiakos. *Web robot detection: A probabilistic reasoning approach*. Computer Networks: The International Journal of Computer and Telecommunications Networking, February, 2009

[3] Tan, P., V. Kumar. *Discovery of Web robot sessions based on their navigational patterns*. Data Mining and Knowledge Discovery, 6, 935, 2002

Notes

[1]http://awstats.sourceforge.net/

[2]Autonomous System (AS) is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet.

[3]This step was taken because the WEKA package does not classify on string attributes. Each attribute was then selected to be of `nominal` type in WEKA, so that each value will be represented as a class instead of numeric.

[4]Weka is a collection of machine learning algorithms for data mining tasks. (http://www.cs.waikato.ac.nz/ ml/weka/)

[5]http://www.csie.ntu.edu.tw/ cjlin/libsvm/