A COMPREHENSIVE STUDY OF THE REGULATION AND BEHAVIOR OF WEB CRAWLERS

A Dissertation in
Information Sciences and Technology
by
Yang Sun

© 2008 Yang Sun

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2008
The dissertation of Yang Sun was reviewed and approved by the following:

C. Lee Giles  
Professor of Information Sciences and Technology  
Dissertation Advisor  
Chair of Committee

James Z. Wang  
Associate Professor of Information Sciences and Technology

Prasenjit Mitra  
Assistant Professor of Information Sciences and Technology

Runze Li  
Associate Professor of Statistics

John Yen  
Professor of Information Sciences and Technology  
Associate Dean of Graduate Programs

*Signatures are on file in the Graduate School.
Abstract

Search engines and many web applications such as online marketing agents, intelligent shopping agents, and web data mining agents rely on web crawlers to collect information from the web, which has led to an enormous amount of web traffic generated by crawlers alone. Due to the unregulated open-access nature of the web, crawler activities are extremely diverse. Such crawling activities can be regulated from the server side by deploying the Robots Exclusion Protocol in a file called robots.txt. Ethical crawlers (and many commercial) will follow the rules specified in robots.txt files. Since the Robots Exclusion Protocol has become a de facto standard for crawler regulation, a thorough study of the regulation and behavior of crawlers with respect to the Robots Exclusion Protocol allows us to understand the impact of search engines and the current situation of privacy and security issues related to web crawlers.

The Robots Exclusion Protocol allows websites to explicitly specify an access preference for each crawler by name. Such biases may lead to a “rich get richer” situation, in which a few popular search engines ultimately dominate the web because they have preferred access to resources that are inaccessible to others. We propose a metric to evaluate the degree of bias to which specific crawlers are subjected. We have investigated 7,593 websites covering education, government, news, and business domains, and collected 2,925 distinct robots.txt files. Results of content and statistical analysis of the data confirm that the crawlers of popular search engines and information portals, such as Google, Yahoo, and MSN, are generally favored by most of the websites we have sampled. The biases toward popular search engines are verified by applying the bias metric to 4.6 million robots.txt files from the web. These results also show a strong correlation between the search engine market share and the bias toward particular search engine crawlers.

Since the Robots Exclusion Protocol is only an advisory standard, actual crawler behavior may differ from the regulation rules. In other words, crawlers may ignore robots.txt files or violate part of the rules in robots.txt files. A thorough analysis of web access logs reveals many potential ethical and privacy issues in web crawler generated visits. We present the log analysis results of three large scale websites and the applications of the data extracted from the log analysis including estimating the crawler population and user stability measures.

To minimize negative aspects of crawler generated visits on websites, the ethical issues of crawler behavior with respect to the crawling rules specified in websites is studied in this thesis. As many web site administrators and policy makers have come to rely on the informal contract set forth by the Robots Exclusion Protocol, the degree to which web crawlers respect robots.txt policies has become an important issue of computer ethics. We analyze the behaviors of web
crawlers in a crawler honeypot, a set of websites where each site is configured with a distinct regulation specification using the Robots Exclusion Protocol in order to capture specific behaviors of web crawlers. A set of ethicality models is proposed to measure the ethicality of web crawlers computationally based on their conformance to the regulation rules. The results show that ethicality scores vary significantly among crawlers. Most commercial web crawlers receive good ethicality scores; however, many commercial crawlers still consistently violate certain robots.txt rules.

The bias and ethicality measurement results calculated based on our proposed metrics are important resources for webmasters and policy makers to design websites and policies. We design and develop BotSeer, a web-based robots.txt and crawler search engine that makes these resources available for users. BotSeer currently indexes and analyzes 4.6 million robots.txt files obtained from 17 million websites as well as three large web server logs and provides search services and statistics of web crawlers for researching web crawlers and trends in Robot Exclusion Protocol deployment and adherence. BotSeer serves as a resource for studying the regulation and behavior of web crawlers as well as a tool to inform the creation of effective robots.txt files and crawler implementations.
# Table of Contents

List of Figures viii

List of Tables x

Chapter 1  Introduction 1
  1.1 Methodology ............................................. 7

Chapter 2  Related Work 9
  2.1 Web Crawlers ............................................. 9
  2.2 Robots Exclusion Protocol ................................. 9
  2.3 The robots.txt Files ....................................... 12
  2.4 Log Analysis ............................................... 14
  2.5 Crawler Behavior .......................................... 16
  2.6 Crawler Ethics ........................................... 17
  2.7 Population Estimation ..................................... 18

Chapter 3  Biases toward Crawlers 20
  3.1 Data Collection ........................................... 20
    3.1.1 Data Sources ......................................... 20
    3.1.2 Crawling for Robots.txt ............................... 20
  3.2 Usage of the Robots Exclusion Protocol ..................... 21
  3.3 Robot Bias ................................................ 23
    3.3.1 The GetBias Algorithm ............................... 23
    3.3.2 Measuring Overall Bias ............................... 25
    3.3.3 Examining Favorability ............................... 25
  3.4 Bias Results ............................................. 26
    3.4.1 History of Bias ..................................... 27
    3.4.2 Search Engine Market vs. Robot Bias ................. 29
    3.4.3 Results on Larger Data Set ........................... 29

Chapter 4  Log Analysis 34
  4.1 Crawler Identification .................................... 35
  4.2 Crawler Traffic Statistics ................................. 36
# Chapter 5 BotSeer System

5.1 DATA

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.1 Robots.txt Files</td>
<td>41</td>
</tr>
<tr>
<td>5.1.2 Web Server Logs</td>
<td>44</td>
</tr>
<tr>
<td>5.1.3 Open Source Crawlers</td>
<td>45</td>
</tr>
</tbody>
</table>

5.2 WEB APPLICATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.1 Robots.txt File Search</td>
<td>45</td>
</tr>
<tr>
<td>5.2.2 Crawler Search</td>
<td>47</td>
</tr>
<tr>
<td>5.2.3 Data Analysis</td>
<td>49</td>
</tr>
<tr>
<td>5.2.3.1 Bias Analysis</td>
<td>50</td>
</tr>
<tr>
<td>5.2.3.2 Dynamic Bias Analysis</td>
<td>51</td>
</tr>
<tr>
<td>5.2.3.3 Robot Generated Log Analysis</td>
<td>52</td>
</tr>
</tbody>
</table>

5.3 DISCUSSION

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
</tr>
</tbody>
</table>

# Chapter 6 Crawler Behavior Analysis

6.1 Models

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.1 Vector Model of Crawler Behavior</td>
<td>56</td>
</tr>
<tr>
<td>6.1.2 Ethicality Metrics</td>
<td>57</td>
</tr>
<tr>
<td>6.1.2.1 Binary Model</td>
<td>57</td>
</tr>
<tr>
<td>6.1.2.2 Probabilistic Model</td>
<td>57</td>
</tr>
<tr>
<td>6.1.2.3 Relative Model</td>
<td>58</td>
</tr>
<tr>
<td>6.1.2.4 Cost Model</td>
<td>58</td>
</tr>
</tbody>
</table>

6.2 Experiments

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.1 Crawler Behavior Test: Honeypot</td>
<td>60</td>
</tr>
<tr>
<td>6.2.2 Results</td>
<td>64</td>
</tr>
<tr>
<td>6.2.2.1 Binary Ethicality</td>
<td>65</td>
</tr>
<tr>
<td>6.2.2.2 Probabilistic Ethicality</td>
<td>65</td>
</tr>
<tr>
<td>6.2.2.3 Relative Ethicality</td>
<td>68</td>
</tr>
<tr>
<td>6.2.2.4 Cost Ethicality</td>
<td>69</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.3 Temporal Ethicality</td>
<td>70</td>
</tr>
<tr>
<td>6.2.4 Compare to Favorability</td>
<td>71</td>
</tr>
</tbody>
</table>

6.3 Applications

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
</tr>
</tbody>
</table>

6.4 Effectiveness of Search Engines

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
</tr>
</tbody>
</table>

# Chapter 7 Estimating Crawler Population

7.0.1 Capture-Recapture Models

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0.1.1 Lincoln-Peterson Model</td>
<td>75</td>
</tr>
<tr>
<td>7.0.1.2 Dependency of Capture Sources</td>
<td>76</td>
</tr>
<tr>
<td>7.0.1.3 Model $M_0$</td>
<td>78</td>
</tr>
<tr>
<td>7.0.1.4 Model $M_h$</td>
<td>79</td>
</tr>
<tr>
<td>7.0.1.5 Model $M_t$</td>
<td>80</td>
</tr>
<tr>
<td>7.0.1.6 Model $M_{th}$</td>
<td>81</td>
</tr>
</tbody>
</table>

# Chapter 8 Conclusions and Future Work

8.1 Future Work

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.1 Bias Analysis</td>
<td>84</td>
</tr>
<tr>
<td>8.1.2 Crawler Ethics</td>
<td>86</td>
</tr>
<tr>
<td>8.1.3 BotSeer Service</td>
<td>87</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 The high-level architecture of a general web crawler system.  
1.2 An example of fetching process in crawlers.  
1.3 The flow of an expected crawler activity.  
3.1 Probability of a website that has robots.txt in each domain.  
3.2 Distribution of robots.txt by domain suffixes. Because of long-tailed distribution, only the top 10 suffixes are shown.  
3.3 Most frequently used robot names in robots.txt files. The height of the bar represents the number of times a robot appeared in our dataset.  
3.4 The distribution of a robot being used.  
3.5 Top 10 and Bottom 10 robots ranked by $\Delta P(r)$, the proportion of the difference between favored and disfavored robots.  
3.6 The search engine market share for 4 popular search engines between 12/05 and 09/06, and $\Delta P$ rating of favorability of these engines.  
3.7 Search engine market share vs. robot bias.  
4.1 Distribution of visits per day from each unique IP address.  
4.2 Distribution of visits per day from each unique IP address.  
4.3 The comparison of crawler visits and user visits as a function of date.  
4.4 The geographical distribution of web crawlers.  
4.5 The geographical distribution of web crawlers named as Googlebot. The blue and red circles point out the well behaved and badly behaved Googlebots respectively.  
5.1 The architecture of BotSeer system.  
5.2 The Homepage of BotSeer system.  
5.3 The architecture of BotSeer crawler.  
5.4 Data selection module between physical storage and applications.  
5.5 Distribution of visits per day from each unique IP address.  
5.6 BotSeer robots.txt search component response to query “botname:msnbot”.  
5.7 The crawler search result page for query “googlebot”.  
5.8 The crawler search result page for query “googlebot”.  
5.9 The geographical distribution of web crawlers that visit CiteSeer. Gray points are the location of crawlers that visit CiteSeer. The blue and red circles point out the well behaved and bad behaved “Googlebot” respectively.  
5.10 Detailed bias analysis of a website.  
5.11 Bias analysis result page of 1,858 named crawlers.  
5.12 Dynamic Bias analysis on the query “Robots Exclusion Protocol”.

8
5.13 The statistics of crawler generated traffic in CiteSeer. ................................. 53
5.14 The crawler generated traffic analysis and monitor. ................................. 54
5.15 Comparison of user visits and crawler visits. The press release shows two dates that BotSeer is reported in news media. ................................. 55

6.1 Site structure and robots.txt file for Honeypot 1. ................................. 61
6.2 Site structure and robots.txt file for Honeypot 2. ................................. 62
6.3 Site structure and robots.txt file for Honeypot 3. ................................. 62
6.4 Site structure and robots.txt file for Honeypot 4. ................................. 62
6.5 Site structure and robots.txt file for Honeypot 5. ................................. 63
6.6 Site structure and robots.txt file for Honeypot 6. ................................. 63
6.7 Site structure and robots.txt file for Honeypot 7. ................................. 63
6.8 Site structure and robots.txt file for Honeypot 8. ................................. 64
6.9 Site structure and robots.txt file for Honeypot 9. ................................. 64
6.10 Temporal changes in ethicality of web crawlers. ................................. 71
6.11 A rule based system to monitor the ethicality of web crawlers. ................................. 72
6.12 Comparison of the effectiveness of Google, Yahoo, MSN and Baidu. ................................. 74

7.1 Set drawing of three capture sources $C_1$, $C_2$ and $C_3$. ................................. 77
7.2 The distribution of population size $N$ with model $M_0$. The maximum likelihood is obtained at $N = 11,844$. ................................. 79
7.3 The simulation of population size $N$ in model $M_h$. ................................. 80
7.4 The simulation of population size $N$ in model $M_t$. ................................. 81
7.5 The simulation of population size $N$ in model $M_{th}$. ................................. 82
List of Tables

1.1 HTTP request. ....................................................... 3
2.1 An example of robots.txt file from http://botseer.ist.psu.edu. ............... 11
2.2 Examples of matching between request URL and Record Path. ................. 12
3.1 The average size (in bytes) and average length (in number of lines) of the collected robots.txt files. ................................................................. 22
3.2 The frequency of Crawl-Delay rules for different robots. ....................... 23
3.3 Top 10 favored and disfavored robots. $\sigma$ is the standard deviation of $\Delta P(r)$. .... 29
3.4 Top 5 favored and disfavored robots on the USA university websites. $\sigma$ is the standard deviation of $\Delta P(r)$. ...................................................... 30
3.5 Top 5 favored and disfavored robots in the Government websites. $\sigma$ is the standard deviation of $\Delta P(r)$. ...................................................... 30
3.6 Top 5 favored and disfavored robots in the newspaper websites. $\sigma$ is the standard deviation of $\Delta P(r)$. ...................................................... 31
3.7 Top 5 favored and disfavored robots in the company websites. $\sigma$ is the standard deviation of $\Delta P(r)$. ...................................................... 31
3.8 Top 5 favored and disfavored robots in the European university websites. $\sigma$ is the standard deviation of $\Delta P(r)$. ...................................................... 32
3.9 Top 5 favored and disfavored robots in the Asian university websites. $\sigma$ is the standard deviation of $\Delta P(r)$. ...................................................... 32
4.1 Statistics for IPs addresses visiting a digital library in one week. .............. 36
6.1 Statistics of the rules in robots.txt files of 2.2 million websites. ............ 61
6.2 Probability of a rule being violated or misinterpreted. ....................... 65
6.3 The binary violation table of web crawlers derived from honeypot access logs. 66
6.4 The full violation table of web crawlers derived from honeypot access logs. 67
6.5 Probability Ethicality of web crawlers. ........................................... 68
6.6 Relative ethicality of web crawlers. ................................................. 68
6.7 Content ethicality scores for crawlers that visited the honeypot. ............ 69
6.8 Access ethicality scores for crawlers that visited the honeypot. ............ 69
6.9 Additional ethical behavior of crawlers. ........................................... 70
6.10 Comparison of crawler probabilistic ethicality and favorability scores. .... 71
6.11 Effectiveness of search engines. ................................................. 73
6.12 Comparison of the effectiveness of Google, Yahoo, MSN and Baidu. ....... 74
Chapter 1

Introduction

Web crawlers (a.k.a. “spiders,” “robots,” “bots” or “harvesters”) are self-acting agents that
navigate around the clock through the hyperlinks of the web, harvesting topical resources with
zero cost in human management [12, 11, 51]. Web crawlers are essential to search engines; without
crawlers, there would probably be no search engines. Web search engines, digital libraries, and
many other web applications such as offline browsers, internet marketing software and intelligent
searching agents heavily depend on crawlers to acquire documents [13, 28, 73]. For example,
Google, Yahoo and MSN crawlers traverse billions of web pages periodically to support the variety
of services provided by these search engines. Shopping bots and price bots bring users discounted
products and price comparisons everyday. There is even a dating bot \(^1\) that provides possible
dating information to users. The crawler functions and activities are extremely diverse because
of the variety of tasks. These functions and activities include not only regular crawls of web pages
for general-purpose indexing and public services, but also different types of unethical functions
and activities such as automatic extraction of email and personal identification information as
well as service attacks. Even general-purpose web page crawls can lead to unexpected problems
for Web servers such as a denial of service attack in which crawlers may overload a website
such that normal user access is impeded. Crawler-generated visits can also affect log statistics
significantly so that real user traffic is overestimated.

A general high-level crawler architecture is shown in Figure 5.1. Multithreaded fetchers
download web pages from the World Wide Web and pass the web pages to a link parser that
extracts hyperlinks (URLs) from these pages. The extracted URLs will then be filtered by a
URL filter that encodes the regulation rules provided by corresponding websites. Valid URLs
will be queued and scheduled for fetchers to download from the World Wide Web. A crawling
cycle includes downloading web pages, extracting and filtering URLs, and generating new tasks.

\(^1\)http://en.wikipedia.org/wiki/Windows_Live_Agents
Therefore, a crawler can traverse the web by following hyperlinks.

The traversal strategy of a crawler can be designed based on specific tasks. The goal of web crawlers is to visit all or part of the nodes (web pages) on the web graph. Thus, the traversal strategy is essentially a graph search algorithm. A general web crawler typically implements a breadth first search (BFS), a depth first search (DFS) or a focused search to traverse the web graph. BFS is a search algorithm that starts with the root node and explores all the nodes connecting to it. For each of these nodes, BFS explores their unexplored connecting nodes until it reaches the goal. DFS is a search algorithm that starts with the root node and explores as far as possible along each branch before backtracking. Both BFS and DFS can make sure web crawlers traverse connected web subgraphs. Each has advantages and disadvantages that make the choice problem dependent, however. BFS crawlers need more space to store all visited pages in each node level than DFS crawlers, which only need to store visited pages in one branch of the web graph. DFS crawlers, however, can be trapped by infinite link loops. BFS crawlers also yield higher quality pages [47].

Focused crawling was brought to attention by both researchers and practitioners in the late 90’s [45]. A focused crawler (a.k.a. topical crawler) attempts to download only web pages that are relevant to predefined topics. In practice, BFS, DFS and focused crawling are all used in web crawler designs for different tasks.

A typical fetching process is illustrated in Figure 1.2. A URL is parsed into three parts: protocol, host and request. The protocol and host are translated to port number and IP address respectively which are used to establish socket connection to the remote web server. When the socket connection is successfully established, the request will be sent to the server to download corresponding document. A typical HTTP request is listed in Table 1.1. In the HTTP request fields, the HTTP method (first line), Host, and connection are required fields in HTTP specification. The rest of the fields are recommended. The User-Agent and Referrer fields, however, are very important to study crawler as well as user behavior and click patterns. After the request
is sent to web server, the server will parse the request and follow five steps to respond to the request: 1. identify connection type; 2. lookup Host configuration; 3. locate requested files in local directories; 4. send response to client; 5. log request.

On account of dramatically increasing web related services, web crawlers have become much more complicated, and their crawling functions and activities vary significantly. As a result, the regulation of web crawlers has become a difficult and important problem. Because of the highly automated nature of the crawlers, rules must be made to regulate their crawling activities in order to prevent an undesired impact to the server workload and to prevent access to information that is not to be offered to the public. In addition, these rules should assist welcome crawlers index the website more efficiently.

The Robots Exclusion Protocol has been proposed [37] to provide advisory regulations for crawlers to follow. A file called robots.txt, which contains crawler access policies, is deployed in the root directory of a website and is accessible to all crawlers. The Robots Exclusion Protocol allows website administrators to indicate to visiting robots which parts of their site should not be visited as well as a minimum time interval between visits. If there is no robots.txt file on a website, robots are free to crawl all content. Ethical crawlers read this file and obey the rules during their visit to the website. An expected crawler activity is illustrated in Figure 1.3. The
expected crawler activity should be parsing the domain from a URL and examine whether the request is restricted by the robots.txt files before fetching web pages.

The robots.txt convention has been adopted by the community since the late 1990s and has continued to serve as one of the predominant means of crawler regulation. Our study shows that more than 30% of active websites deploy this standard to regulate crawler activities [70]. Although the robots.txt convention has become a de facto standard for crawler regulation, little work has been done to investigate its usage in detail, especially on the scale of the Web.

More importantly, as websites may favor or disfavor certain crawlers by assigning to them different access policies, this bias can lead to a “rich get richer” situation whereby some popular search engines are granted exclusive access to certain resources, which in turn could make them even more popular. Considering the fact that users often prefer a search engine with broad (if not exhaustive) information coverage, this “rich get richer” phenomenon may introduce a strong influence on users’ choice of search engine, which will eventually be reflected in the search engine market share. On the other hand, it is often believed (although this is an exaggeration) that “what is not searchable does not exist,” and this phenomenon may also introduce a biased view of the information on the Web.

The robots.txt files also play an important role in indexing digital repositories over OAI-PMH protocol [42]. The OAI-PMH describes a set of verbs and specifications for harvesters to request interoperable metadata of digital objects from online digital repositories [16]. A digital repository (data provider) typically provides the metadata of its digital objects to service providers (e.g., search engines) with a set of base URLs through which OAI-PMH requests are submitted. The metadata of a digital object also contains the URL where the actual object can be located.
Major search engines including Google and Yahoo already built harvesters to collect resources from large repositories [42]. Other efforts have been made to convert the metadata of digital objects to general web pages for general web crawlers to index [77].

Both the OAI-PMH base URLs and the URLs of the digital objects are regulated by robots.txt files since major search engine crawlers follow the robots.txt rules whenever they are fetching a URL. In such situation, the robots.txt files may prevent the harvesters from requesting resources from digital repositories. Thus, improper design of robots.txt rules can lead to unexpected results. It is reported that robots.txt files protecting both the OAI-PMH base URLs and the object URLs which results in that the major search engines fail to index the repositories even though the intent of implementing OAI-PMH for the repositories is to share their metadata [42]. Therefore, the robots.txt file is very important to the search engine coverage of digital repositories. The research of search engine coverage of the OAI-PMH corpus suggests that robots.txt files in digital repositories should be carefully designed so that it does not prevent service providers from requesting the repositories over OAI-PMH and also provides necessary protections at the same time.

Alternative techniques are used to regulate the behavior of Web crawlers. The Robots META tag can be written in an HTML page to prevent ethical crawlers from crawling or indexing the page. Similar to the Robots Exclusion Protocol, the Robots META tag is an advising rule that will only be obeyed by ethical crawlers. Enforcing techniques such as robot traps\(^2\) or IP restrictions are also used in Web servers. These techniques can usually keep unwelcome crawlers away from the site. Although the enforcing techniques seem more powerful in regulating crawlers, they are also vulnerable if the new crawling techniques can avoid the traps [27]. In addition to attempts to restrict web crawlers, efforts have been made to assist crawlers to better index websites. A new tool, Sitemaps [61], helps search engines by specifying how often pages are changed. However, the Sitemap protocol does not yet include specifications of load balancing for websites.

Although the Robots Exclusion Protocol has been widely adopted by websites, it is an advisory standard that cannot prevent misbehaved crawlers. How web crawlers really behave on websites is the key to designing regulation rules and standards to guide and regulate web crawlers. According to our research, a significant portion of web crawlers fails to identify themselves. A significant amount of crawler visits also violate the rules specified in robots.txt. This diverse crawler behavior brings up ethical issues related to automated computer software as well.

Previous research on computer ethics (a.k.a. machine ethics) has primarily focused on how humans use technology. With the growing role of autonomous agents on the internet, however, the ethics of machine behavior has come to the attention of the research community [2, 4, 24, 32, 53, 73]. The field of computer ethics is especially important in the context of web crawling since collecting and redistributing information from the web often leads to considerations of information privacy and security.

Because the Robots Exclusion Protocol serves only as an unenforced advisory to crawlers,\(^2\)http://www.fleiner.com/bots/
web crawlers may ignore the rules and access part of the forbidden information on a website. It is reported that many web crawlers, including some commercial crawlers, disregard part of the rules or disregard robots.txt completely. One company even states that it does not follow regulation standards in its crawler description documents. Crawlers can attempt to acquire hidden documents by guessing URLs from restricted directory names listed in the robots.txt files. The crawler visit interval is also important to websites, especially when the access bandwidth is limited. Webmasters complain about crawlers occupying too much bandwidth because users experience delay in regular access of websites. Clearly, the usage of the Robots Exclusion Protocol and the behavior of web crawlers with respect to the robots.txt rules provide a foundation for a quantitative measure of web crawler ethics.

It is difficult to interpret ethicality in different websites. The unethical actions in one website may not be considered unethical in others. For example, crawling more than 1 page per second may significantly affect the normal usage of a small website. The same crawling speed, however, may be expected for a large website with fast updating content. Little research has been done in the area of developing computational models to formulate ethicality and to provide measures of web crawler ethics.

This thesis studies the regulation and behavior of web crawlers based on the Robots Exclusion Protocol and web access log analysis. The contributions of the thesis are as follows:

- The thesis proposes a quantitative metric to measure the bias in the regulations of web crawlers. By applying the metric to a large sample of websites, the thesis presents the findings about the most favored and disfavored crawlers and suggests complements to the Robots Exclusion Protocol.
- The thesis analyzes the statistics of crawler traffic for different websites and estimates the population of active web crawlers with capture-recapture models.
- The thesis formally defines crawler ethicality based on common concepts of ethical crawlers in the research community and proposes a vector space model of web crawler ethics based on the Robots Exclusion Protocol. A honeypot is designed to capture crawler behaviors. The ethicality results of major search engine crawlers that visited our honeypot are presented in this thesis.
- A robots.txt and web crawler search engine, BotSeer, is designed and developed to provide resources for analyzing web crawler behavior and trends in the Robots Exclusion Protocol deployment.

---

4 http://www.munax.com/crawlingfaq.htm
6 http://www.webmasterworld.com/forum30/33074.htm
7 http://botseer.ist.psu.edu
1.1 Methodology

The goal of the thesis is to conduct a comprehensive survey of web crawler regulations and behavior based on the Robots Exclusion Protocol. The widely adopted Robots Exclusion Protocol provides a foundation for the study. As introduced in previous chapters, web crawlers may be regulated by different rules in the robots.txt files. Such bias may have a significant impact on the indexable content of each crawler. Since the Robots Exclusion Protocol specifies a set of advisory rules, the crawler may behave differently than the regulation rules specify. Ethical issues arise with crawler behavior violations.

This thesis attempts to explore the following questions:

- How is the Robots Exclusion Protocol used?
- Does a robot bias exist?
- How should such a bias be measured quantitatively?
- What are the implications for such a bias?
- How many crawlers are there on the web?
- How do crawlers actually behave on the web?
- How can we measure a crawler’s ethicality towards a website?

A large-scale data collection is the basis for answering these questions and for showing the significance of the quantitative analysis. For this thesis, 4.6 million robots.txt files were collected and 200GB of web access logs were processed to support the quantitative analysis.

The study of crawler regulation is focused on the analysis of the bias in robots.txt files. In the first step, we collect real-world robots.txt files from 17 million unique websites with different functionalities, covering the domains of education, government, news and business. Thus, regulation bias can be compared across domains. A quantity metric is proposed based on the Robots Exclusion Protocol to measure its biases in terms of web crawlers. A formal definition of bias in robots.txt files is proposed, and an overall favorability metric is proposed to measure the bias toward each web crawler over a set of websites. With the favorability metric, the most favored and disfavored crawlers on the Web are identified.

To summarize the analysis of the robots.txt files and crawler behavior and to provide information to the public, we design and develop BotSeer, the first robots.txt and crawler search engine to deliver data and measures to the public.

To study the behavior of web crawlers, we process 200GB of the access logs from three independent websites. Statistics from the crawler-generated logs provide answers to the fifth questions above. We also set up a honeypot, a set of websites designed based on the specifications in the Robots Exclusion Protocol and common rules derived from 4.6 million sample websites.

---

8http://botseer.ist.psu.edu/
The honeypot includes all cases where the robots.txt rules can be violated by crawlers. Once a crawler visits the honeypot, its behaviors are recorded and analyzed. With the honeypot design and data, we proposed a set of models to measure the ethicality of web crawlers. Thus, ethical issues can be explored by examining the ethicality measure of web crawlers. The most ethical and unethical crawlers are identified based this measure.

The Robots Exclusion Protocol will be discussed in detail in Chapter 2. Chapter 3 discusses the methodology of the thesis in the study of web crawlers. Chapter 4 presents the study of usage and the bias analyses of robots.txt files. Chapter 5 presents the statistics of web crawler traffic and log analysis results. Chapter 6 presents BotSeer, the first robots.txt and crawler search engine to assist crawler related research and regulation policy design. The crawler behavior study and ethicality measures are presented in Chapter 7 with the honeypot, a set of websites to test the behavior of web crawlers, and presents the measurement of crawler ethicality. Chapter 8 presents the capture-recapture models used to estimate web crawler population.
Chapter 2

Related Work

2.1 Web Crawlers

2.2 Robots Exclusion Protocol

The Robots Exclusion Protocol\(^1\) (REP) is a convention that allows website administrators to indicate to visiting crawlers which parts of their site should not be visited. If there is no robots.txt file on a website, crawlers are free to crawl all content. However, other types of restrictions may apply to crawlers, such as Terms of Services of websites. It is the crawler administrator’s responsibility to check all possible restrictions.

The format of the Robots Exclusion Protocol is described in [37]. A file named “robots.txt” with Internet Media Type “text/plain” is placed under the root directory of a Web server. Each line in the robots.txt file has the format:

\[
<\text{field}>:<\text{optionalspace}><\text{value}><\text{optionalspace}>.
\]

In the original specification, there are three types of case-insensitive directives for the \(<\text{field}>\) to specify the rules: User-Agent, Allow and Disallow. Another unofficial directive, Crawl-Delay, is also used by many websites to limit the frequency of crawler visits.

- **User-Agent**: followed by a string value \(str_{\text{crawler}}\) indicates that any crawler whose name is a superstring of \(str_{\text{crawler}}\) should follow the rules below. If there is more than one match, only the first one will be honored. For example, “User-Agent: google” applies to all Google crawlers such as “Googlebot”, “googlebot-image” and “mediapartners-google”. If \(str_{\text{crawler}}\) is “*”, it means the rules apply to all crawlers without regard to any specific match.

\(^1\)\url{http://www.robotstxt.org/norobots-rfc.txt}
• **Allow:** and **Disallow:** followed by a string value \texttt{str\_directory} inform a crawler which directory it is allowed or disallowed to visit. A crawler must match each of the \texttt{str\_directory} values against the URL (in the order \texttt{str\_directory} occur in the record if there are more than one) and use the first match found. If no match is found, the default assumption is that the URL is allowed. Only one directory or file per line is allowed, and regular expressions are not supported. The directory can be a full path or a partial path; for the **Disallow** directive, any URL that starts with this directory will not be retrieved. If a %xx encoded octet (e.g., %3D means “=”) is encountered, it is unencoded prior to comparison unless it is the “/” character, which has special meaning in a path. For example, **Disallow:** /help disallows both /help.html and /help/index.html, whereas **Disallow:** /help/ would disallow /help/index.html but allow /help.html.

An interesting case occurs when \texttt{str\_directory} following **Allow** : or **Disallow** : is blank. The meaning of this rule is ambiguous: Does it mean “nothing is disallowed/allowed” or “nothing matches this line”?

• **Crawl-Delay:** followed by a number value \texttt{num\_delay} tells a crawler how long it should wait between two consecutive visits. The delay time is measured in seconds. For example, **Crawl-Delay:** 20 indicates that the crawler should wait for at least 20 seconds before attempting to visit the website again.

The **robots.txt** file starts with one or more **User - Agent** fields, specifying which crawlers the rules apply to, followed by a number of **Disallow** : and/or **Allow** : fields indicating the actual rules to regulate the crawler. The matching order is specific in REP: “...A crawler must attempt to match the paths in Allow and Disallow lines against the URL, in the order they occur in the record. The first match found is used.” [37] Comments are allowed anywhere in the file and consist of optional whitespaces. Comments are started with a comment character ‘#’ and terminated by the linebreak.

A sample **robots.txt** is listed in Table 2.1 (this robots.txt file is from **BotSeer**\textsuperscript{2}):

The robots.txt file should be interpreted as noting that **Googlebot, MSNBot, Sharp** and **Teoma** cannot visit /robotstxtanalysis, /uastring and /whois and that the minimum interval between two visits should be greater than 600 seconds. **BotSeer** crawler can visit any directory and file on the server without any delay. All other crawlers should follow the rules under **User - Agent** : * and cannot visit the directories and files matching /robots/, /src/, /botseer, /uastring, /srcseer, /robotstxtanalysis or /whois, and the minimum interval between two visits should also be larger than 600 seconds.

The REP specifies how crawlers should react in situations where robots.txt files are temporarily unavailable and are forbidden:

• On server response indicating access restrictions (HTTP Status Code 401 or 403), a crawler should regard access to the site as completely restricted.

\textsuperscript{2}http://botseer.ist.psu.edu/robots.txt
Table 2.1. An example of robots.txt file from http://botseer.ist.psu.edu.

- On a request attempt that resulted in a temporary failure, a crawler should defer visits to the site until such time as the resource can be retrieved.

- On a server response indicating Redirection (HTTP Status Code 3XX), a crawler should follow the redirects until a resource can be found.

It is also specified in the REP that user agent names should be matched regardless of case. %xx encoded octet should be decoded prior to examining restriction rules unless it is a “/”
character. Substring matching should say that any request URL starting with the rule values should follow the rule command. Table 2.2 shows some examples of how request URL should match the rule value (Record Path).

<table>
<thead>
<tr>
<th>Record Path</th>
<th>URL path</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tmp</td>
<td>/tmp</td>
<td>yes</td>
</tr>
<tr>
<td>/tmp</td>
<td>/tmp.html</td>
<td>yes</td>
</tr>
<tr>
<td>/tmp</td>
<td>/tmp/a.html</td>
<td>yes</td>
</tr>
<tr>
<td>/tmp/</td>
<td>/tmp</td>
<td>no</td>
</tr>
<tr>
<td>/tmp/</td>
<td>/tmp/a.html</td>
<td>yes</td>
</tr>
<tr>
<td>/a%3cd.html</td>
<td>/a%3cd.html</td>
<td>yes</td>
</tr>
<tr>
<td>/a%3Cd.html</td>
<td>/a%3cd.html</td>
<td>yes</td>
</tr>
<tr>
<td>/a%3cd.html</td>
<td>/a%3Cd.html</td>
<td>yes</td>
</tr>
<tr>
<td>/a%3Cd.html</td>
<td>/a%3Cd.html</td>
<td>yes</td>
</tr>
<tr>
<td>/a%2fb.html</td>
<td>/a%2fb.html</td>
<td>yes</td>
</tr>
<tr>
<td>/a%2fb.html</td>
<td>/a.html</td>
<td>no</td>
</tr>
<tr>
<td>/a/b.html</td>
<td>/a%2fb.html</td>
<td>no</td>
</tr>
<tr>
<td>/a/b.html</td>
<td>/a/b.html</td>
<td>yes</td>
</tr>
<tr>
<td>/%7ejoe/index.html</td>
<td>/joe/index.html</td>
<td>yes</td>
</tr>
<tr>
<td>/%7ejoe/index.html</td>
<td>/%7ejoe/index.html</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 2.2. Examples of matching between request URL and Record Path.

There are ambiguities and missing specifications in the original REP. The rule `Disallow [:emptystring]` can be understood as matching everything or nothing. Does the rule mean to allow crawlers to crawl anything or nothing? Case sensitivity issues are not addressed for URL path matching. The REP also does not specify how to deal with errors and conflicts.

Another issue regards the “Crawl-Delay” field. Since this field is not in the original Robots Exclusion Protocol, not every crawler recognizes this rule. How should webmasters design robots.txt if they do not know whether a crawler recognizes the rule or not? There is no further discussion in the Robots Exclusion Protocol about these issues.

Failure to use proper authentication or other restriction may result in exposure of restricted information. It is even possible that the occurrence of paths in the /robots.txt file may expose the existence of resources not otherwise linked to on the site, which may aid people in guessing URLs [37]. Since the REP is neither an official standard nor an enforcement standard, the consequences and costs of disobeying robots.txt rules are not clear.

2.3 The robots.txt Files

The Robots Exclusion Protocol describes a set of rules to regulate the web crawlers. More specifically, certain web pages can be hidden (restricted) from web crawlers by the Robots Exclusion Protocol. The regulation standard brings many issues to the accessibility and searchability of
the web. Research on the invisible web shows that the Robots Exclusion Protocol is an important factor of keeping web pages being indexed by search engines [59]. The robots.txt rules also restricts information that is ...some of the best resources available on the Web exists in this form and to overlook it would be to detract from the full value of the Web [52]. The study of the invisible web brings a great attention to the information society. Much research has been following the directions of exploring the search engines invisible web and the implications [17, 40, 44, 76]. However, these papers typically examine the invisible web from a high-level perspective that discusses and reviews the issues. Quantitative analysis to measure the invisible part of the web is ignored.

With the increase of the impact of search engines and their crawlers, the usage of the Robots Exclusion Protocol brings attention to some researchers. A study of the usage of robots.txt in U.K. universities and colleges investigated 163 websites and 53 robots.txt files [34]. Robots.txt files are considered as a standard that showing Robots the Door. The research takes a quantitative approach to study the usage of the robots.txt files. Robots.txt files were examined in terms of file size (in bytes and as total number of lines), usage of comments, disallowed directives, and regulated user-agent lines within U.K. university domains. Results show that the majority robots.txt files are small and contain fewer than 12 directives. Although the sample size is small, there are still errors being found in the collection. The study only gives a brief introduction to crawler regulations in a small sample of websites. The results could be strongly biased toward a specific type of websites (universities) and/or a specific region (U.K.). Thus, it fails to provide an in-depth analysis of the usage and implications for the web and search engine crawlers in general.

The usage of robots.txt has been studied as an aid for indexing to protect information [22]. Sixty samples from Fortune Global 500 company websites were manually examined in this work, showing that “robots.txt files are not widely used by the sampled group and for most of the sites on which they appear, they are redundant. ...they exclude robots from directories which are locked anyway.” The percentage of websites uses robots.txt files to prevent certain directories and files being access by crawlers is given for 2000 (17-23%) and 2001 (23-37%). Although the number shows that a significant amount of websites uses robots.txt files to regulate crawlers, the sample size (60) is too small the support the findings. Another important factor that is not considered in this research is that most commercial crawlers agree to follow the robots.txt rules. The agreement gives the Robots Exclusion Protocol a strong power in crawler regulation. The study is also biased by a specific type of websites (Fortune 500 company websites). Our investigation shows a contrary result that may be due to the difference in sample size, domain and time.

The legal aspects of obeying robots.txt is discussed in [7] where agents and bots are distinguished by human intervention. As they quoted: A robot is a program that automatically traverses the Web’s hypertext structure by retrieving a document, and recursively retrieving all documents that are referenced. Normal Web browsers are not robots, because they are operated by a human, and don’t automatically retrieve referenced documents (other than online images), an intelligent agent is treated as a well trained program whose behavior is authorized by human users. Therefore, the search agents are excluded from the regulation rules according to the pa-
per. The study also indicates that the General Terms and Conditions of a website is a contract between the users and the websites. If obeying robots.txt rules are explicitly mentioned in the contract, violating them may face legal consequences. The General Terms and Conditions, however, should be accessible for all agents. According to our study of the usage of robots.txt files, the policy makers of robots.txt rules are clearly considering search agents as robots and specify rules for them. This is a contradiction to the explanation of the search agents. This thesis adopts a well accepted concept of web crawlers/robots in the research community [67, 71]. Search agents as well as other well trained intelligent agents and crawlers are all considered crawlers and are expected to be regulated by the Robots Exclusion Protocol.

Much crawler related research has mentioned the regulation of crawlers in terms of obeying the robots.txt rules [13, 30]. Many commercial search engines also include on their websites the description of their crawlers that indicates how the crawlers will respect the robots.txt rules.

However, none of the aforementioned work investigates the content of robots.txt in terms of biases towards web crawlers. In addition, the sample sizes of previous studies have tended to be relatively small considering the size of the Web.

### 2.4 Log Analysis

Web server access logs are the most important resources used to study web crawler behavior, as these logs have detailed information for every visit generated by crawlers as well as by users. Thus, log analysis technologies (a.k.a. log mining) are very important in our crawler behavior study. A web access log record includes the connection IP address, the time and date of access, the requested files (URL), the identity of a crawler or a browser, and the status code of the HTML response from the server. A web log mining process typically includes three phases [66]: (1) data preprocessing, (2) pattern discovery, and (3) pattern analysis. From web access logs, information such as clicking sequence and visiting frequency can then be used by the web application to infer users’ motivations and goals as well as crawlers' logic.

Extracting the crawler-generated log records from access logs is a complicated problem since not all crawlers identify themselves as such, instead pretending to be normal, user-driven browsers. Much research has been done in the area of web usage mining [5, 14, 46, 65, 66], which mentions crawler identification in the log data preprocessing step. Heuristic rules based on indicators of non-human behavior are used in [5]. The indicators are: (a) the repeated request for the same URL from the same host; (b) a time interval between requests too short to apprehend the contents of a page; and (c) a series of requests from one host all of whose referer URLs are empty. The referer URL of a request is empty if the URL was typed in, requested using a bookmark, or requested using a script [5]. Crawlers are recognized through their non-human behavior, such as depth-first traversal of a whole directory or repeated accesses to the same page every 30 seconds in [65]. It

---

3http://www.google.com/support/webmasters/bin/answer.py?answer=70897&cbid=-17992dek38mcr&src=cb&lev=topic
is also mentioned in [65] that robots are identified by their deviation from typical human-user patterns in the SurfAID project of IBM. The crawler identification techniques are based on the assumption that crawler behavior is significantly different than human behavior. Since most of the above research was conducted before 2000, neither spambot nor pretending-to-be-other-bot issues in these papers are as serious problems as they are today. None of the work deals with the crawler identification problem carefully in terms of accuracy of identifying crawlers. Even recent work still follows the simplified steps to identify web crawlers [72].

Crawler generated log brings a wider attention to the research community in recent years. Much research has been done in the area of crawler detection and identification [25, 29, 67, 71]. A detailed survey of crawler generated traffic is presented in [71] which motivates the research of crawler identification. In the paper, the navigational patterns in the clickstream data is used to identify web crawlers. First, sessions with features are extracted from the clickstream data (web access logs) by a heuristic algorithm. A web crawler model is proposed to classify sessions based on its features. The results show that the model accuracy of detecting crawlers is more than 90% in a university department website. The classification features in the proposed model including requests for robots.txt, HEAD requests and unassigned referrers are very informative and are also used in this thesis. Another web crawler detection research applies the content type of server files to identify web crawlers [29]. In this approach, the crawlers can be detected when the visits include different content types. The accuracy of the proposed algorithms is tested on a small scale website and receives good results. However, the successfulness of this approach strongly depends on the type of website. It is hard to generalized the method to other websites. A more sophisticated model is proposed in [67] where a Bayesian network classifier is built based on six variables. The six variables include: a) Maximum sustained click-rate. b) Session duration. c) Percentage of image requests. d) Percentage of pdf/ps requests. e) Percentage of requests with 4xx response code. f) robots.txt file requests. A human expert is asked to manually classify the sessions extracted from the web access logs to test the Bayesian networks trained on different data sets. The result shows the Bayesian approach can correctly detect more than 80% of crawlers in all cases. One problem for these crawler detection approaches is the accuracy of the test data. User-Agent field in the request header is not mentioned as an important feature or variable to identify crawlers. The field is assume to be used as a criterion for human experts to judge whether a session is a crawler session or human user session. However, human experts may not be able to identify crawlers with fake identities (pretended to be browsers). In practice, it is very hard for human to identify fake browsers without a knowledge of the different distribution of user sessions and robot sessions. An overview and evaluation of the crawler detection technologies are presented in [25].

\footnote{See surfaid.dfw.ibm.com}
2.5 Crawler Behavior

Prior research has been conducted that studies crawler behaviors using web access logs [3, 18, 21, 62, 63]. [3] analyzes the characterization of workload generated by web crawlers and studies their impact on caching. A hierarchical approach with three levels: session, function and request is used to characterize the workload of requests generated by robots. A crawler generated workload model has been developed to help balancing online bookstores for a better performance. The results show that crawlers cause a significant increase in the miss ratio of a server side cache, resulting in higher costs for request responses [3]. The analysis results also suggest websites should be designed to provide different services for crawlers and users which will result in a more efficient caching mechanism for both crawlers as well as users.

[18] characterizes and compares the behavior of five search engine crawlers including Google, Inktomi, AltaVista, FastSearch and CiteSeer. A set of metrics is also proposed to describe qualitative characteristics of crawler behavior including three categories: HTTP message, retrieved resources and temporal behavior. Format preference metric helps to distinguish the crawlers that prefer a certain format of resources. Frequency of visits metric measures the frequency of a crawler visiting a website. It is interesting to observe that there is no correlation between this metric and the popularity or geographical location of a website. Coverage metric is defined to measure the exhaustiveness of a crawler's visit on a website. The metrics can also be used in automatic crawler detection, improving crawler design and reduce impact on web server performance. The result analysis produces seven major findings which include: 1. Crawler activity has a noticeable impact on Web-server workload; 2. Crawler generated requests have higher percentage of GET requests; 3. Crawlers generate more error code than other clients; 4. Crawlers fetch web pages more aggressively than other clients; 5. The distribution of the frequency of crawler requests across different resources (resource popupularity) is not Zipf. 6. HTTP response to crawler requests exhibit a high variability in size. 7. Crawler visits show a periodic pattern.

A functional classification scheme is proposed to analyze crawler traffic in [21]. Crawler functions are classified into ten categories including Indexer, Experimental, Verifier, RSS crawler, Analyzer, Harvester, Scraper, IRCBot, Anonymous and Unknown. Hierarchical criteria are used to classify crawlers into the ten categories including three rules 1. undocumented crawlers are likely to be ill-constructed with high risk. 2. Crawler should fit in the category whose functionality is more specific. 3. Crawlers that request multiple types of resources rank higher. The results on the log analysis of a university department website show that crawlers in different categories have significant different behavior. Unfortunately, the paper did not provide detailed analysis and results on the behavior of crawlers in different categories. The list of sample crawler for each category is also missing.

The change of web crawler behavior in decaying a website is studied in [62]. A website is designed with randomly generated HTML and PDF files. The total number of files on the test website are decaying from 954 to 0 during 90 days. The plot of number of pages visited by crawler as a function of days shows how crawlers adjust their behavior according to the changes
of websites. The results show that many crawlers adapt to site changes, adjusting their behavior accordingly. Except for their home-made crawler, Google crawler’s response is almost the same as the decaying curve. Yahoo and MSN do not crawl the site as frequently as Google. The research provides some interesting behavior change of crawlers. It is hard to conclude, however, what is the reason for the behavior changes, since a crawler visits may be determined by many factors including in-links and the popularity of the website.

The crawler behavior at deep and wide websites has been studied in [63]. Four websites are designed by randomly generated content from English-language literature with 100 hyperlinks on the homepage and 100 levels deep. Two linking styles, Buffet style and Bread Crumb style, are used to organize the pages on the four websites. Buffet style has a index page of all 100 links for each level. Bread Crumb style has only one entry link to the homepage of each level. Each style is setup under a .com domain and a .edu domain. The statistics show a significant difference of the crawler visits in different styles. The comparison on different crawlers also show that Google crawler reaches new website faster than Yahoo and MSN crawlers. Google crawler also has a much higher worst-coverage (99%) than Yahoo (3%) and MSN (2.5%).

### 2.6 Crawler Ethics

One major topic of this thesis is to measure the ethical issues brought about by crawler activities. Much research has discussed the ethical issues related to computers and the web [4, 10, 32, 33, 41]. The theoretical foundation for machine ethics is discussed by [4]. Prototype systems are implemented to provide advice on ethically questionable actions. It is expected that the behavior of more fully autonomous machines, guided by this ethical dimension, may be more acceptable in real-world environments [4]. Social contract theory is used to study computer professionals and their social contract with society [32]. Privacy and piracy issues of software are discussed in [10]. The need for informed consent in Web related information research has been advocated in [41] and debated in [33]. The subject of the aforementioned work, however, is the general use of intelligent machines instead of a targeted investigation of web crawlers.

The ethical issues of the design and administration of web crawlers are discussed in [23]. It provides guidelines for ethical crawlers to follow. It is one of the earliest papers that discuss the ethics of crawlers. It suggests that the first rule for any software agents is to follow the Robots Exclusion Protocol. In addition, an agent should identify themselves, open their purpose and functionality to the public if they provide related service, moderate the visiting frequency, respect the regulations and restrict unanticipated requests. The paper also foresees the increase of crawler traffic in the future. Since it is written in 1995, many new issues of crawler ethics remain untouched. However, it can still be considered as a high-level guideline toward designing an ethical crawler.

The ethical factors are examined in a more recent study [73]. Four types of ethical issues that web crawlers may raise for society and individuals are discussed: denial of service, cost, privacy, and copyright. Denial of service refers to the crawler behavior that generates repeated
visits in a very short time interval so that normal user access is impeded. Cost refers to the bandwidth occupied by crawlers that incur cost of bandwidth to the owners of websites. *Web information may invade privacy if it is used in certain ways, principally when information is aggregated on a large scale over many Web pages.* As an example, spam list can be generated by crawling emails from websites which obviously violates the privacy. Copyright issues may raise when crawlers try to keep permanent copies of web content. An ethical crawl guideline is described for crawler owners to follow. This guideline suggests taking legal action or initiating a professional organization to regulate web crawlers. However, the research does not provide measures to characterize crawler ethicality or evaluate the consequences of unethical behavior.

Since many web crawlers are built for web mining projects, the ethical issues of crawlers also relate to the web mining ethics. The ethical issues in web data mining is studied in [74]. Web content and structure mining and web usage mining are examined separately for privacy and individuality. The legal issues surrounding network measurements have also been discussed recently in [60]. Since much web data mining research involves crawling, the privacy issues are also ethical issues concerning the crawlers used by researchers. The papers provides some guidelines to consider ethical issues.

None of the above mentioned work, however, provides a quantitative measure of the ethical factors (ethicality) of web crawlers.

### 2.7 Population Estimation

The above reviewed research shows that web crawlers become a major contributor to web traffic. With the development of computer technologies, building a web crawler becomes a relatively easy task. There are hundreds of open source web crawler projects available online as well as homemade web crawlers developed for specific services and research projects. One fundamental question investigated in this thesis is: how many web crawlers are there?

Capture-recapture (CR) models have a rather long history in the biometry literature where they have been used to estimate the population sizes of wildlife animals. The simplest CR method, the Lincoln-Peterson method, has been used to estimate unknown population since 1894. Since the assumptions in Lincoln-Peterson model are often not held, advanced models using maximum likelihood estimators are proposed in the research community [57, 48, 54, 55]. A framework of applying maximum likelihood method to fit capture recapture data is first proposed in [48]. Different assumptions are made to develop models for different estimation type. The models include $M_0$ (no variation exists), $M_t$ (allows capture probability to vary by time), $M_b$ (allows behavioral response to captures), $M_h$ (allows heterogenous individual capture probabilities), as well as the combination of models $M_{th}$, $M_{tb}$, $M_{bh}$ and $M_{tbh}$. The unified Maximum Likelihood estimates for closed CR Models are presented in [54] using mixtures. The likelihood function for each model is given along with examples of applying models. The capture recapture

---

6http://en.wikipedia.org/wiki/Mark_and_recapture
experiment design is discussed in [55]. More recently, CR models for open population have been
developed [9, 43].

CR models are also used to estimate the size of the web [6, 8, 20, 38, 39] and telephone lines
[56]. Lincoln-Peterson model has been used in [38, 39] to estimate the size of the web by the
overlap of results from different search engines. [6] also conducts a similar study but estimates
the size by minimizing the sum of the squared differences of the estimated overlaps between pairs
of search engines. The previous research [6, 8, 38, 39] has been compared in [20]. The Rasch
model is also applied to propose a more accurate estimation in [20]. [56] presents a Bayesian
approach and uses Markov chain Monte Carlo methods to obtain inference from the posterior
distributions of the telephone line universe. The research fits the telephone log data to the
maximum likelihood CR model $M_h$ to estimate the parameters. In this thesis, we uses a similar
approach as [56] and adopt the maximum likelihood estimation framework [48] to estimate the
population of web crawlers.
Chapter 3

Biases toward Crawlers

3.1 Data Collection

To observe the potential robot bias on the Web, this thesis studies a wide range of websites with different domains and from different physical locations. The data collection is described below in detail.

3.1.1 Data Sources

The Open Directory Project [19] is the largest and most comprehensive human-maintained Web directory. Our primary source to collect the initial URLs to feed our crawler is DMOZ because the Open Directory Project classifies a large URL collection into different categories. It enables us to collect data from different domains and physical locations. Our collection from the Open Directory Project covers three domains: education, news, and government. The university domain is further broken down into the American, European, and Asian university domains.

Since the directory structure of the business domain in DMOZ is complicated and has a significant amount of overlaps, we use the 2005 Fortune Top 1000 Company List [1] as our data source.

There are certain limitations inherent in our data collection. First, because the website collection in DMOZ is limited for other countries especially for non-English websites, the majority of the websites are from the USA. Second, because the DMOZ entries are organized by human editors, there might be errors. Finally, we collect business websites from the Fortune 1000 list which contains data mostly of large corporations, so that the data in that domain may not be representative of small businesses. We intend to address these limitations in future research.

3.1.2 Crawling for Robots.txt

We have implemented a specialized focused crawler for this study. The crawler starts by crawling the metadata of a website obtained from DMOZ including the functional classification, the name of the website, and the physical location of its affiliated organization. Then, the crawler checks the
existence of robots.txt for that domain and downloads existing robots.txt files for offline analysis. A parsing and filtering module is also integrated into our crawler to eliminate duplicates and for ensuring that retrieved pages are within the target domain.

Besides the root level directory, our crawler also examines other possible locations of the robots.txt file. The subdirectories of a website (up to level 3) are inspected. Results show that there are few cases where robots.txt is not placed under the root directory where it should be according to the Robots Exclusion Protocol. Misspelled filenames are also examined by our crawler. In rare cases the filename “robot.txt” (which will be ignored by robots) is used instead of “robots.txt”.

In order to observe the temporal properties, the crawler has performed 5 crawls for the same set of websites from Dec. 2005 to Oct. 2006. In order to analyze the temporal properties, the downloaded robots.txt files are archived according to the date of the crawl.

3.2 Usage of the Robots Exclusion Protocol

Statistics: We crawled and investigated 7,593 unique websites including 600 government websites, 2,047 newspaper websites, 1,487 USA university websites, 1,420 European university websites, 1,039 Asian university websites, and 1,000 company websites.

![Figure 3.1. Probability of a website that has robots.txt in each domain.](image)

Overall, the percentage of websites that have robots.txt has increased from 35% to 38.5% in the past 11 months (see Figure 3.1). Since search engines and intelligent searching agents become more important for accessing web information, this result is expected. The Robots Exclusion Protocol is more frequently adopted by government (44%), newspaper (46%) and university websites in the USA (45.9%). It is used extensively to protect information not to be offered to the public and balance workload for these websites.

There are 1056 named robots found in our dataset. The universal robot “*” is the most frequently used robot in the User-Agent field and used 2744 times, which means 93.8% of robots.txt files have rules for the universal robots. 72.4% of the named robots appeared only once or twice.
The downloaded robots.txt files are also categorized by the domain suffixes of the URLs from which they are found (see Figure 3.2). Since the suffix of a website’s domain name does not always correctly represent the category of the website, we only use the DMOZ categorization for the rest of our study.

**Figure 3.2.** Distribution of robots.txt by domain suffixes. Because of long-tailed distribution, only the top 10 suffixes are shown

**Size and Length:** An interesting observation is that the sizes and lengths of the robots.txt files on governmental websites are significantly larger than those from the other investigated domains. There are 26 files at a length of 68 lines and 4 at 253 lines. A reasonable explanation is that government websites tend to adopt more sophisticated robot restrictions, which results in larger and longer robots.txt files (see Table 3.1).

<table>
<thead>
<tr>
<th>Domain</th>
<th>avg size with standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA edu</td>
<td>625.1 ±158.1</td>
</tr>
<tr>
<td>European edu</td>
<td>422.5 ±86.7</td>
</tr>
<tr>
<td>Asian edu</td>
<td>270.1 ±108.0</td>
</tr>
<tr>
<td>Business</td>
<td>895.8 ±472.6</td>
</tr>
<tr>
<td>Gov</td>
<td>1551.2 ±760.6</td>
</tr>
<tr>
<td>News</td>
<td>509.7 ±41.3</td>
</tr>
</tbody>
</table>

Table 3.1. The average size (in bytes) and average length (in number of lines) of the collected robots.txt files.

**Crawl Delay:** The field name “Crawl-Delay” in robots.txt files has recently been used by web administrators. Web server administrators most likely use this field in their robots.txt files to arrange an affordable workload. The usage of Crawl-Delay increased from 40 cases (1.5%) in Dec. 2005 to 140 cases (4.8%) in Oct. 2006. The frequency of Crawl-Delay rules for different robots are shown in Table 3.2.

**Incorrect Use:** When we examine the content of the collected robots.txt files, a significant number of incorrect uses of the Robots Exclusion Protocol has been found. These incorrect uses include misnamed files, incorrect locations, and conflicting rules. Because of these incorrect uses, the access policy will be ignored by robots. We observe 13 cases of misnamed robots.txt files and find 23 files in which a specific name such as “crawler”, “robot”, or “webcrawlers” appear
<table>
<thead>
<tr>
<th>Robot Name</th>
<th>Number of Delay Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>msnbot</td>
<td>42</td>
</tr>
<tr>
<td>slurp</td>
<td>36</td>
</tr>
<tr>
<td>yahooseeker/cafekelsa</td>
<td>12</td>
</tr>
<tr>
<td>googlebot</td>
<td>7</td>
</tr>
<tr>
<td>teoma</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3.2. The frequency of Crawl-Delay rules for different robots.

in the User-Agent field. General name in the User-Agent field is an incorrect use of the Robots Exclusion Protocol. We also found 282 robots.txt files with ambiguous rules and 18 files with conflicting rules (e.g. a directory is disallowed first and then allowed or allowed first and then disallowed).

The actual method for how robots will access the robots.txt is not specified in the Robots Exclusion Protocol. Open source crawlers such as “Websphinx”, “Jspider” and “Nutch” checks the robots.txt file right before crawling each URL by default. We observe “Googlebot”, “Yahoo! Slurp” and “MSNbot” cache the robots.txt files for a website. During the modification of robots.txt file, these robots might disobey the rules. As a result, there are a few disallowed links appearing in these search engines.

Comments In Robots.txt: We have found cases in which comments in robots.txt files are not written for version or explanation but written for users or robot administrators¹. Even a blog has been found in a robots.txt file².

3.3 Robot Bias
We propose $\Delta P(r)$, a measure of the favorability of robots across a sample of robots.txt files, to measure the degree to which specific robots are favored (or disfavored) by a set of websites. A formal definition of the robot bias (favored or disfavored) is described below.

3.3.1 The GetBias Algorithm
Our definition of a favored robot is a robot allowed to access more directories than the universal robot according to the robots.txt file in the website. The universal robot is any robot that has not matched any of the specific User-Agent names in the robots.txt file. In other words, the universal robot represents all the robots that do not appear by name in the robots.txt file.

Let $F$ be the set of robots.txt files in our dataset. Given a robots.txt file $f \in F$, let $R$ denote the set of named robots for a given robots.txt file $f$. For each named robot $r \in R$, We define the $GetBias(r, f)$ algorithm as specified in Algorithm 1. $GetBias$ measures the degree to which a named robot $r$ is favored or disfavored in a given robots.txt file $f$.

Let $DIR$ be the set of all directories that appear in a robots.txt file $f$ of a specific website. $DIR$ is used as an estimation of the actual directory structure in the website because the Robot

¹http://www.ebay.com/robots.txt
²http://www.webmasterworld.com/robots.txt
Algorithm 1 GetBias(r, f)

1: if r is * then
2: return 0
3: end if
4: Construct DIR for f;
5: bias = 0
6: for all d ∈ DIR do
7: if d is allowed for * then
8: \( D_u \leftarrow d \)
9: end if
10: end for
11: for all d ∈ DIR do
12: if d is allowed for r then
13: \( D_r \leftarrow d \)
14: end if
15: end for
16: bias = |\( D_r \)| − |\( D_u \)|
17: return bias

Exclusion Protocol considers any directory in the website that does not match the directories in the robots.txt as an allowed directory by default. \( D_u \in DIR \) is the set of directories that the universal robot "*" is allowed to visit. If there are no rules specified for User-Agent: *, the universal robot can access everything by default. \( D_r \in DIR \) is the set of directories that a given robot r is allowed to visit. |\( D_u \)| and |\( D_r \)| are the number of directories in \( D_u \) and \( D_r \).

For a given robot r, the algorithm first counts how many directories in DIR are allowed for r. Then it calculates the bias score for robot r as the difference between the number of directories in DIR that are allowed for the robot r and the number of directories that are allowed for the universal robot. In the GetBias algorithm, the bias of the universal robot is treated as the reference point 0 (GetBias returns 0). The bias scores of favored robots returned by GetBias are positive values. Higher score of a robot means the robot is more favored. On the contrary, the bias scores of disfavored robots returned by GetBias are negative values, which is consistent with our bias definition. Thus, the bias of a robot in a robots.txt file can be represented by a categorical variable with three categories: favored, disfavored, and no bias.

As an example, consider the robots.txt file in http://BotSeer.ist.psu.edu from Section 2: DIR = \{"/robots/", "/src/", "/botseer", "/uastring", "/robotstxtanalysis", "/whois"\}. According to the algorithm we have \( D_u = \{null\} \), \( D_{botseer} = \{"/robots/", "/src/", "/botseer", "/uastring", "/robotstxtanalysis", "/whois"\} \) and \( D_{google} = \{"/robots/", "/src/", "/botseer", "/uastring", "/robotstxtanalysis", "/whois"\} \). Thus, |\( D_u \)| = 0, |\( D_{botseer} \)| = 7, and |\( D_{google} \)| = 5. According to Algorithm 1, bias\(_u\) = |\( D_u \)| − |\( D_u \)| = 0, bias\(_{botseer}\) = |\( D_{botseer} \)| − |\( D_u \)| = 7 and bias\(_{googlebot}\) = |\( D_{google} \)| − |\( D_u \)| = 5. Thus, the robots “googlebot” and “botseer” are favored by this website, and they are categorized as favored. All other robots will be categorized as no bias.
3.3.2 Measuring Overall Bias

Based on the bias score for each file, we propose $\Delta P(r)$ favorability in order to evaluate the degree to which a specific robot is favored or disfavored on a set of robots.txt files. Let $N = |F|$ be the total number of robots.txt files in the dataset. The $\Delta P(r)$ favorability of a robot $r$ can be defined as below:

$$\Delta P(r) = P_{\text{favor}}(r) - P_{\text{disfavor}}(r)$$

$$= \frac{N_{\text{favor}}(r) - N_{\text{disfavor}}(r)}{N}. \quad (3.1)$$

where $N_{\text{favor}}(r)$ and $N_{\text{disfavor}}(r)$ are the number of times a robot is favored and disfavored respectively. $P_{\text{favor}}(r)$ is the proportion of the robots.txt files in which a robot $r$ is favored; $P_{\text{disfavor}}(r)$ is the proportion of the robots.txt files in which a robot $r$ is disfavored.

The proportions of robots.txt files that favor or disfavor a specific robot are simple measures for survey statistics; however, in our dataset the two proportions in isolation are not very accurate in reflecting the overall biases in our sample since there are more than two events (favor, disfavor and no bias). This means that $P_{\text{favor}}(r) + P_{\text{disfavor}}(r) < 1$. Each event only reflects one aspect of the bias. For example, a robot named “ia_archiver” is favored by 0.24% of the websites in our dataset and the proportion of sites that favor “momspider” is 0.21%. Alternatively, the proportions of sites that disfavor “ia_archiver” and “momspider” are 1.9% and 0%, respectively. If we only consider the favored proportion, we will reach the conclusion that “ia_archiver” is more favored than “momspider”.

$\Delta P(r)$ is the difference of the proportions of sites that favor and disfavor a specific robot, and thus treats both cases in unison. For the above example $\Delta P(\text{ia_archiver})$ is -1.66% and $\Delta P(\text{momspider})$ is 0.21%. Thus, “momspider” is more favored than “ia_archiver”. For any no-bias robot $r$, $\Delta P(r)$ is 0. The bias measure can eliminate the misleading cases and still be intuitively understandable (favored robots have positive numbers and disfavored robots have negative numbers).

3.3.3 Examining Favorability

The favorability is actually a ranking function of robots. To evaluate accuracy of this ranking function, we run a ranking performance test based on Kendall’s rank correlation method [35]. The rank correlation method is briefly described below. The details of the ranking performance evaluation using partial order can be found in [31].

For a robots.txt file $f$, let $m_a$ be a bias measure function for all robots $R$ appearing in $f$. Let $r_i$ and $r_j \in R$ be two named robots in $f$. We denote $r_i <_{m_a} r_j$ if $r_i$ is ranked higher than $r_j$ for measure $m_a$. Thus, for any two measure functions $m_a$ and $m_b$, Kendall’s $\tau$ can be defined based on the number $P_f$ of concordant pairs and the number $Q_f$ of discordant pairs. A pair $r_i \neq r_j$ is concordant if both $m_a$ and $m_b$ agree in how they order $r_i$ and $r_j$. It is discordant if they disagree.
In this case, Kendall’s τ can be defined as:

\[ \tau_f(m_a, m_b) = \frac{P_f - Q_f}{P_f + Q_f} \]  

(3.2)

For any given measure \( m_a \) and \( m_b \), the \( \tau_f(m_a, m_b) \) represents how well the two ranking measures agree with each other in a file \( f \). Let \( m_a \) represent the actual ranking function of robots. Although we do not know the actual ranking function, we have the partial ranking of robots for each robots.txt file based on the bias score defined previously. Thus, computing the \( \tau_f(m_a, m_b) \) for all robots.txt files will show how well the measure \( m_b \) agrees with \( m_a \) for the actual ranking of robots in file \( f \).

We calculate \( \tau_f(m_a, m_b) \) for each robots.txt files \( f \) in our dataset. If \( \tau_f(m_a, m_b) = 1 \) for a given robots.txt file, we consider that the file \( f \) is a concordant file for \( m_a \) and \( m_b \). Otherwise, the file \( f \) is a discordant file. By counting the concordant files \( P \) and discordant files \( Q \) in the dataset, we can compute the average \( \tau(m_a, m_b) \). Note that \( P + Q = N \), thus,

\[ \tau(m_a, m_b) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{N} \]  

(3.3)

We rank the robots using the \( \delta P \) favorability. The ranked lists are then compared with the actual ranking using the method introduced above. The average \( \tau \) value is 0.957 which we believe is accurate enough to measure the overall bias of a robot.

### 3.4 Bias Results

There are 1056 named robots found in our dataset. The universal robot “\*” is the most frequently used robot in the User-Agent field and used 2744 times, which means 93.8% of robots.txt files have rules for the universal robots. 72.4% of the named robots appeared only once or twice. The most frequently appearing robots in our dataset are shown in Figure 3.3.

![Figure 3.3. Most frequently used robot names in robots.txt files. The height of the bar represents the number of times a robot appeared in our dataset.](image-url)
3.4.1 History of Bias

The distribution of how many times a robot is used (see Figure 3.4) did not change significantly over the past 11 months. Thus, we show the bias results from the latest crawl since not much has changed.

![Figure 3.4. The distribution of a robot being used.](image)

Since most of the robots appeared only once or twice in the dataset, their ranking scores are ranked in the middle of the list and are almost indistinguishable. We consider only the top ranked (favored) and bottom ranked (disfavored) robots. The 10 most favored robots and 10 most disfavored robots are shown in Figure 3.5.

The 10 most favored robots and 10 most disfavored robots are also listed in Table 3.3 where $N$ is the sample size, $N_{favor}$ is the number of times the robot is favored, $N_{disfavor}$ is the number of times the robot is disfavored and $\sigma = \sqrt{\Delta P(r)(1-\Delta P(r)) / N}$ is the categorical standard deviation\[49\] of $\Delta P(r)$. The categorical standard deviation $\sigma$ gives the variance when using $\Delta P$ to estimate the favorability of robots on the Web.

Our bias measure shows that the most highly favored robots are from well-known search engines and organizations, e.g., “Google”, “Yahoo” and “MSN” are favored much more than the remaining robots. Please note that for some robots in the disfavored category, their $\Delta P$ favorability does not show a significant difference due to their rare appearances in the sampled robots.txt files.

On the other hand, most of the disfavored robots are email collectors (“CherryPicker” and “emailsiphon”) and off-line browsers (“Wget” and “webzip”). From the privacy perspective, it is reasonable for webmasters to exclude robots whose major purpose is to collect private information. Also, webmasters typically do not want their websites to be copied entirely by others. However, even robots from well-known companies can be disfavored e.g., “MSIECrawler” (Microsoft) and “ia_archiver” (Alexa). “MSIECrawler” is a robot embedded in Internet Explorer (IE). When IE users bookmark a page while offline, MSIECrawler downloads the page and all links related to it, including links, images, JavaScript and Style sheets, when the user is next online. “ia_archiver” is the crawler from archive.org and Alexa.com. A list of detailed description
Figure 3.5. Top 10 and Bottom 10 robots ranked by \( \Delta P(r) \), the proportion of the difference between favored and disfavored robots.

\( \Delta P(r) \) is also computed for other domains including the government, newspaper, and company domains, as well as European and Asian university domains. The results of \( \Delta P(r) \) as well as the number of times a robot is favored and disfavored are shown for these domains respectively in Table 3.5, 3.6, 3.7, 3.8, and 3.9. Due to the long-tailed distribution, even the top most disfavored robots do not have a significant difference in \( \Delta P(r) \) for most domains.

We find that robot biases in different domains vary significantly. Google is always the most favored robot. Other top favored robots vary in different domains. Yahoo (“slurp” is a Yahoo robot) and MSN are also favored in most domains, but they are not significantly favored over other robots. Other top favored robots are mostly open source crawlers and crawlers from well-known organizations. Disfavored robot lists vary widely for different domains. Most of these robots are still email collectors and offline browsers. The differences could be due to the different behaviors of robots in different domains (e.g., emailsiphon may crawl business websites more often than others to collect business contacts).

Of known robots appeared in this paper can be found on the web\(^3\).

\[^3\text{http://botseer.ist.psu.edu/namedrobots.html}\]
Table 3.3. Top 10 favored and disfavored robots. $\sigma$ is the standard deviation of $\Delta P(r)$.

| Favored Robots (Sample size $N = 2925$) |
|-----------------|-------|-------|
| robot name      | $N_{favor}$ | $N_{disf}$ | $\Delta P(r)$ | $\sigma$ |
| google          | 877    | 25     | 0.2913         | 0.0084   |
| yahoo           | 631    | 34     | 0.2041         | 0.0075   |
| msn             | 349    | 9      | 0.1162         | 0.0059   |
| scooter         | 341    | 15     | 0.1104         | 0.0058   |
| lycos           | 91     | 5      | 0.0294         | 0.0031   |
| netmechanic     | 84     | 10     | 0.0253         | 0.0029   |
| httlig          | 15     | 3      | 0.0041         | 0.0012   |
| teoma           | 13     | 3      | 0.0034         | 0.0011   |
| oodlebot*       | 8      | 0      | 0.0027         | 0.0010   |
| momspider       | 6      | 0      | 0.0021         | 0.0008   |

| Disfavored Robots (Sample size $N = 2925$) |
|-----------------|-------|-------|
| robot name      | $N_{favor}$ | $N_{disf}$ | $\Delta P(r)$ | $\sigma$ |
| msiecrawler     | 0      | 85     | -0.0291        | 0.0031   |
| ia_archiver     | 7      | 55     | -0.0164        | 0.0023   |
| cherrypicker    | 0      | 37     | -0.0126        | 0.0021   |
| emailsiphon     | 3      | 34     | -0.0106        | 0.0019   |
| roverbot        | 2      | 27     | -0.0085        | 0.0017   |
| psbot           | 0      | 23     | -0.0079        | 0.0016   |
| webzip          | 0      | 21     | -0.0072        | 0.0016   |
| wget            | 1      | 22     | -0.0072        | 0.0016   |
| linkwalker      | 2      | 20     | -0.0062        | 0.0015   |
| asterias        | 0      | 18     | -0.0062        | 0.0015   |

3.4.2 Search Engine Market vs. Robot Bias

In order to study the impact of the “rich get richer” effect, we calculate the correlation between the robot bias and the search engine market share for specific companies. The market share of Google, Yahoo, MSN and Ask in the past 11 months and the $\Delta P$ favorability for the corresponding robots are considered two independent variables (see Figure 3.6).

The Pearson product-moment correlation coefficient[64] (PMCC) between the two variables is a measure of the tendency of two variables $X$ and $Y$ measured on the same object or organism to increase or decrease together. For our dataset, the Pearson correlation of the market share of the four companies and the $\Delta P(r)$ of their corresponding robots is 0.930 with P-Value < 0.001. The search engine market share 4 and robot bias in September, 2006 is shown in Figure 3.7.

3.4.3 Results on Larger Data Set

The above results are analyzed based on a small sample of websites in order to obtain biases for different domains. We also collect the robots.txt files from a large set of 4.6 million websites.

---

4http://www.netratings.com
### Favored Robots (Sample size $N = 683$)

<table>
<thead>
<tr>
<th>Robot Name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>googlebot</td>
<td>91</td>
<td>9</td>
<td>0.1201</td>
<td>0.0124</td>
</tr>
<tr>
<td>momspider</td>
<td>4</td>
<td>0</td>
<td>0.0056</td>
<td>0.0029</td>
</tr>
<tr>
<td>slurp</td>
<td>8</td>
<td>6</td>
<td>0.0029</td>
<td>0.0021</td>
</tr>
<tr>
<td>linklint</td>
<td>2</td>
<td>0</td>
<td>0.0029</td>
<td>0.0021</td>
</tr>
<tr>
<td>msnbot</td>
<td>4</td>
<td>2</td>
<td>0.0029</td>
<td>0.0021</td>
</tr>
</tbody>
</table>

### Disfavored Robots (Sample size $N = 683$)

<table>
<thead>
<tr>
<th>Robot Name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cherrypicker</td>
<td>0</td>
<td>14</td>
<td>-0.0205</td>
<td>0.0054</td>
</tr>
<tr>
<td>emailsiphon</td>
<td>1</td>
<td>11</td>
<td>-0.0146</td>
<td>0.0046</td>
</tr>
<tr>
<td>teleport</td>
<td>0</td>
<td>7</td>
<td>-0.0102</td>
<td>0.0038</td>
</tr>
<tr>
<td>wget</td>
<td>0</td>
<td>7</td>
<td>-0.0102</td>
<td>0.0038</td>
</tr>
<tr>
<td>webzip</td>
<td>0</td>
<td>7</td>
<td>-0.0102</td>
<td>0.0038</td>
</tr>
</tbody>
</table>

Table 3.4. Top 5 favored and disfavored robots on the USA university websites. $\sigma$ is the standard deviation of $\Delta P(r)$.

### Favored Robots (Sample size $N = 264$)

<table>
<thead>
<tr>
<th>Robot Name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>21</td>
<td>4</td>
<td>0.0644</td>
<td>0.0151</td>
</tr>
<tr>
<td>teoma</td>
<td>3</td>
<td>0</td>
<td>0.0114</td>
<td>0.0065</td>
</tr>
<tr>
<td>ultraskeek</td>
<td>4</td>
<td>2</td>
<td>0.0076</td>
<td>0.0053</td>
</tr>
<tr>
<td>momspider</td>
<td>2</td>
<td>0</td>
<td>0.0076</td>
<td>0.0053</td>
</tr>
<tr>
<td>yahoo</td>
<td>9</td>
<td>8</td>
<td>0.0038</td>
<td>0.0038</td>
</tr>
</tbody>
</table>

### Disfavored Robots (Sample size $N = 264$)

<table>
<thead>
<tr>
<th>Robot Name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>mj12bot</td>
<td>0</td>
<td>5</td>
<td>-0.0189</td>
<td>0.0084</td>
</tr>
<tr>
<td>turnitinbot</td>
<td>0</td>
<td>4</td>
<td>-0.0152</td>
<td>0.0075</td>
</tr>
<tr>
<td>linkwalker</td>
<td>0</td>
<td>4</td>
<td>-0.0152</td>
<td>0.0075</td>
</tr>
<tr>
<td>webstripper</td>
<td>0</td>
<td>3</td>
<td>-0.0114</td>
<td>0.0065</td>
</tr>
<tr>
<td>ia_archiver</td>
<td>1</td>
<td>4</td>
<td>-0.0114</td>
<td>0.0065</td>
</tr>
</tbody>
</table>

Table 3.5. Top 5 favored and disfavored robots in the Government websites. $\sigma$ is the standard deviation of $\Delta P(r)$.

The results confirm our bias analysis on the small sample. The data is available on BotSeer.\(^5\)

\(^5\)http://botseer.ist.psu.edu
### Favored Robots (Sample size $N = 942$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>690</td>
<td>4</td>
<td>0.728</td>
<td>0.0145</td>
</tr>
<tr>
<td>yahoo</td>
<td>595</td>
<td>16</td>
<td>0.615</td>
<td>0.0159</td>
</tr>
<tr>
<td>msn</td>
<td>334</td>
<td>3</td>
<td>0.3514</td>
<td>0.0156</td>
</tr>
<tr>
<td>scooter</td>
<td>328</td>
<td>1</td>
<td>0.3471</td>
<td>0.0155</td>
</tr>
<tr>
<td>lycos</td>
<td>79</td>
<td>0</td>
<td>0.0839</td>
<td>0.0090</td>
</tr>
</tbody>
</table>

### Disfavored Robots (Sample size $N = 942$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>msiecrawler</td>
<td>0</td>
<td>74</td>
<td>-0.0786</td>
<td>0.0088</td>
</tr>
<tr>
<td>ia_archiver</td>
<td>0</td>
<td>33</td>
<td>-0.035</td>
<td>0.0060</td>
</tr>
<tr>
<td>roverbot</td>
<td>1</td>
<td>20</td>
<td>-0.0202</td>
<td>0.0046</td>
</tr>
<tr>
<td>propsmart</td>
<td>0</td>
<td>18</td>
<td>-0.0191</td>
<td>0.0045</td>
</tr>
<tr>
<td>psbot</td>
<td>0</td>
<td>8</td>
<td>-0.0085</td>
<td>0.0030</td>
</tr>
</tbody>
</table>

Table 3.6. Top 5 favored and disfavored robots in the newspaper websites. $\sigma$ is the standard deviation of $\Delta P(r)$.

### Favored Robots (Sample size $N = 339$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>25</td>
<td>2</td>
<td>0.0678</td>
<td>0.0137</td>
</tr>
<tr>
<td>yahoo</td>
<td>8</td>
<td>2</td>
<td>0.0177</td>
<td>0.0072</td>
</tr>
<tr>
<td>msn</td>
<td>5</td>
<td>0</td>
<td>0.0147</td>
<td>0.0065</td>
</tr>
<tr>
<td>ask</td>
<td>3</td>
<td>0</td>
<td>0.0088</td>
<td>0.0051</td>
</tr>
<tr>
<td>ultraceek</td>
<td>1</td>
<td>0</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
</tbody>
</table>

### Disfavored Robots (Sample size $N = 339$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailsiphon</td>
<td>0</td>
<td>7</td>
<td>-0.0206</td>
<td>0.0077</td>
</tr>
<tr>
<td>cherrypicker</td>
<td>0</td>
<td>6</td>
<td>-0.0177</td>
<td>0.0072</td>
</tr>
<tr>
<td>ia_archiver</td>
<td>1</td>
<td>7</td>
<td>-0.0177</td>
<td>0.0072</td>
</tr>
<tr>
<td>inspiderguy</td>
<td>0</td>
<td>6</td>
<td>-0.0177</td>
<td>0.0072</td>
</tr>
<tr>
<td>wget</td>
<td>0</td>
<td>6</td>
<td>-0.0177</td>
<td>0.0072</td>
</tr>
</tbody>
</table>

Table 3.7. Top 5 favored and disfavored robots in the company websites. $\sigma$ is the standard deviation of $\Delta P(r)$. 
### Favored Robots (Sample size $N = 537$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>44</td>
<td>2</td>
<td>0.0782</td>
<td>0.0116</td>
</tr>
<tr>
<td>ultraseek</td>
<td>8</td>
<td>0</td>
<td>0.0149</td>
<td>0.0052</td>
</tr>
<tr>
<td>yahoo</td>
<td>7</td>
<td>0</td>
<td>0.0130</td>
<td>0.0049</td>
</tr>
<tr>
<td>htdig</td>
<td>5</td>
<td>0</td>
<td>0.0093</td>
<td>0.0041</td>
</tr>
<tr>
<td>msn</td>
<td>3</td>
<td>0</td>
<td>0.0056</td>
<td>0.0032</td>
</tr>
</tbody>
</table>

### Disfavored Robots (Sample size $N = 537$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cherrypicker</td>
<td>0</td>
<td>8</td>
<td>-0.0149</td>
<td>0.0052</td>
</tr>
<tr>
<td>emailsiphon</td>
<td>1</td>
<td>6</td>
<td>-0.0093</td>
<td>0.0041</td>
</tr>
<tr>
<td>emailcollector</td>
<td>4</td>
<td>7</td>
<td>-0.0074</td>
<td>0.0037</td>
</tr>
<tr>
<td>turnitinbot</td>
<td>0</td>
<td>3</td>
<td>-0.0055</td>
<td>0.0032</td>
</tr>
<tr>
<td>emailwolf</td>
<td>0</td>
<td>3</td>
<td>-0.0055</td>
<td>0.0032</td>
</tr>
</tbody>
</table>

**Table 3.8.** Top 5 favored and disfavored robots in the European university websites. $\sigma$ is the standard deviation of $\Delta P(r)$.

### Favored Robots (Sample size $N = 160$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>10</td>
<td>1</td>
<td>0.0563</td>
<td>0.0182</td>
</tr>
<tr>
<td>ndl</td>
<td>4</td>
<td>0</td>
<td>0.025</td>
<td>0.0123</td>
</tr>
<tr>
<td>yahoo</td>
<td>2</td>
<td>0</td>
<td>0.0125</td>
<td>0.0088</td>
</tr>
<tr>
<td>msn</td>
<td>1</td>
<td>0</td>
<td>0.0063</td>
<td>0.0063</td>
</tr>
<tr>
<td>scooter</td>
<td>1</td>
<td>0</td>
<td>0.0063</td>
<td>0.0063</td>
</tr>
</tbody>
</table>

### Disfavored Robots (Sample size $N = 160$)

<table>
<thead>
<tr>
<th>robot name</th>
<th>$N_{favor}$</th>
<th>$N_{disfavor}$</th>
<th>$\Delta P(r)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>msiecrawler</td>
<td>0</td>
<td>2</td>
<td>-0.0125</td>
<td>0.0088</td>
</tr>
<tr>
<td>teleport</td>
<td>0</td>
<td>2</td>
<td>-0.0125</td>
<td>0.0088</td>
</tr>
<tr>
<td>webzip</td>
<td>0</td>
<td>2</td>
<td>-0.0125</td>
<td>0.0088</td>
</tr>
<tr>
<td>wget</td>
<td>0</td>
<td>2</td>
<td>-0.0125</td>
<td>0.0088</td>
</tr>
<tr>
<td>larbin</td>
<td>0</td>
<td>2</td>
<td>-0.0125</td>
<td>0.0088</td>
</tr>
</tbody>
</table>

**Table 3.9.** Top 5 favored and disfavored robots in the Asian university websites. $\sigma$ is the standard deviation of $\Delta P(r)$.
Figure 3.6. The search engine market share for 4 popular search engines between 12/05 and 09/06, and $\Delta P$ rating of favorability of these engines.

Figure 3.7. Search engine market share vs. robot bias.
Log Analysis

Web server access logs are one of the most important resources used to study web crawler behavior, as these logs have detailed information for every visit generated by crawlers as well as by users. A typical web access log record includes the connection IP address, the time and date of access, the requested files (URL), the identity of a crawler or a browser, and the status code of the HTML response from the server (see Figure 4.1). A typical Web log mining process includes three phases [66]: (1) data preprocessing, (2) pattern discovery, and (3) pattern analysis. From web access logs, information such as clicking sequence and visiting frequency can then be used by the web application to infer users’ motivations and goals as well as crawlers’ logic.

Extracting the crawler-generated log records from access logs is a complicated problem since not all crawlers identify themselves as such, instead pretending to be normal, user-driven browsers. Much research has been done in the area of log mining [5, 14, 46, 65, 66], which mentions crawler identification in the log data preprocessing step. Since most of the above research was conducted before 2000, neither spambot nor pretending-to-be-other-bot issues in these papers are as serious problems as they are today. None of the work deals with the crawler identification problem carefully in terms of accuracy of identifying crawlers.

Figure 4.1. Distribution of visits per day from each unique IP address.
4.1 Crawler Identification

In this thesis, a multi-step log filter is implemented to identify web crawlers. The first step checks the user-agent string field recorded in the access logs. A crawler user-agent string list\footnote{http://user-agents.org/} is used to identify obvious crawlers. The second step involves a more sophisticated log mining method that extracts a set of features to characterize the browsing behavior of an IP address based on sessions. The sessions are identified by correlating the request and reference fields with page structures in websites.

The behavior features include visits per day, the series of visiting times, and session length (time spent on a session). Figure 5.5 shows the distribution of visits of each unique IP address per day for CiteSeer. It shows a power-law distribution for less than 35 visits connecting with another distribution for larger numbers of visits. There are four types of visits to a web server: single user visits, crawler visits, multiple users using the same computer and multiple users visiting via one proxy. It is easy to separate most of the single users from crawlers by setting a visit threshold at 60. But it is hard to distinguish the other types of visits from crawlers.

The reference section in log records is typically sent by web browsers for tracking or caching services for users. Since crawlers do not need such functions, the reference section in crawler requests is typically set to "-". Therefore, the visiting patterns for crawlers typically do not have a hierarchical structure even if the requested pages have such a structure in the website. This type of visiting pattern can be used to identify some crawlers that are hiding their identities.

We do find smart crawlers trying to simulate user browsing behavior. This type of crawler is especially hard to identify with the mixture of the logs of multiple users using a proxy. In such
Table 4.1. Statistics for IP addresses visiting a digital library in one week.

<table>
<thead>
<tr>
<th></th>
<th>Website1</th>
<th>Website2</th>
<th>Website3</th>
<th>Website4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique IPs</td>
<td>256,559</td>
<td>240,089</td>
<td>8,881</td>
<td>2,289</td>
</tr>
<tr>
<td>Crawler</td>
<td>5,336 (2.08%)</td>
<td>5,499 (2.29%)</td>
<td>3,496 (39.36%)</td>
<td>1,250 (54.61%)</td>
</tr>
<tr>
<td>User</td>
<td>251,223 (97.92%)</td>
<td>234,590 (97.71%)</td>
<td>5,385 (60.64%)</td>
<td>1,039 (45.39%)</td>
</tr>
<tr>
<td>Total Visits</td>
<td>3,859,136</td>
<td>5,591,787</td>
<td>219,827</td>
<td>21,270</td>
</tr>
<tr>
<td>Crawler</td>
<td>2,750,761 (71.28%)</td>
<td>1,778,001 (31.80%)</td>
<td>179,755 (81.77%)</td>
<td>13,738 (64.59%)</td>
</tr>
<tr>
<td>Violating REP</td>
<td>305,632 (11.11%)</td>
<td>N/A</td>
<td>11,041 (6.14%)</td>
<td>1,319 (9.6%)</td>
</tr>
<tr>
<td>User</td>
<td>1,108,402 (28.72%)</td>
<td>3,813,786 (68.2%)</td>
<td>40,072 (18.23%)</td>
<td>7,532 (35.41%)</td>
</tr>
</tbody>
</table>

situations, the average session length and the variance of session length are used to distinguish multiple users with one proxy from crawlers.

With a slight training on a naive Bayesian classifier with the above-mentioned features, we are able to identify more than 90% of crawler generated logs from a large scale web server.

4.2 Crawler Traffic Statistics

Web crawlers have become important traffic consumers for most websites. The statistics of web crawler visits show the importance of measuring the ethicality of crawlers. We analyze crawler-generated access logs from four different types of websites, including a large scale academic digital library, an e-commerce website, an emerging small-scale vertical search engine, and a testing website. The detailed statistics for each website are listed in Table 4.1.

The access statistics show that more than 50% of the web traffic is generated by web crawlers on average and that web crawlers occupy more bandwidth than human users on certain types of websites. Although the portion of crawler-generated traffic varies for different types of websites, the contribution of crawler traffic to the overall website traffic is non-negligible. Crawler traffic is especially important for emerging websites. Figure 5.5 compares the crawler visits and user visits for an emerging specialized search engine as a function of date. Each point in Figure 5.5 corresponds to the total visits in that month. There are two dates (2007/06 and 2007/10) associated with news releases of the website. The comparison shows that user responses to news releases are much faster than crawler responses. Crawlers are more likely to revisit the website after they find it, however.

The crawler traffic statistics also check whether each crawler visit is successfully regulated by the robots.txt file. The results show that up to 11% of crawler visits violated or misinterpreted the robots.txt file.

The User-Agent field is typically embedded in the HTTP request header according to HTTP standards. This field is used for the webmasters to identify crawlers. There are crawlers intentionally specifying themselves as web browsers.² The geographical distributions of “fake browsers” (crawlers pretending to be browsers) are shown in Figure 4.4. The distribution shows that these web crawlers are not limited to specific geographical areas. In fact, each day of logs indicates that

²http://www.munax.com/crawlingfaq.htm
Figure 4.3. The comparison of crawler visits and user visits as a function of date.

"Fake browsers" (crawlers pretended to be browsers)

Figure 4.4. The geographical distribution of web crawlers.

the geographical distribution of crawlers is worldwide, which implies that it is not possible to block them according to their geographical location. Furthermore, crawlers are not only “faking” browsers. Commercial crawlers’ identities have also been used by other crawlers. For example, 46 crawlers named themselves as Googlebot although their IP addresses cannot be associated with Google by reverse DNS lookup in one-day’s CiteSeer access log (see Figure 5.9). We use blue to indicate Googlebots that obey all robots.txt rules and red to indicate fake Googlebots.

The analysis of web crawler generated traffic shows that web crawlers become an important
Figure 4.5. The geographical distribution of web crawlers named as Googlebot. The blue and red circles point out the well behaved and badly behaved Googlebots respectively.

part of web usage.
BotSeer System

BotSeer is designed to provide an efficient tool for researchers, webmasters and developers to study web crawler related issues and design websites. The purpose of BotSeer is to build an information system that assists the regulation and development of web crawlers.

The Robots Exclusion Protocol is a de facto standard on the Web to regulate the behavior of web crawlers. Websites can explicitly specify an access preference for each crawler by name in their robots.txt files. Such biases data can be a valuable resource to the research of Web crawler regulation and more. For example, a new type of network can be constructed based on the similarity of two websites in terms of bias toward web crawlers. Such network may present new perspectives of the Web graph based on similar privacy concerns and hidden community of webmasters or web designers. Thus, robots.txt files are an important data source for BotSeer.

BotSeer is designed by considering the following possible use cases:

- It is always important to know how well a crawler is received on the Web. Since webmasters can specify the preference of favor or disfavor a given crawler in the robots.txt files, the question of how well a crawler is received can be answered by studying the robots.txt files [69].

- People are interested in studying the crawler regulations in specific domains \(^1\). (In this case, the regulation in government websites are of particular interest.)

- The bias toward different crawlers over the web gives an indication of crawler accessibility. If the compared crawlers are search engine crawlers, it can be used as an evidence of which search engine is more favored by a certain group of websites.

- Our previous study shows that there is a strong correlation between the favorability of a web crawler and its search engine market share[69]. It is possible to relate the favorability of web crawlers to their behavior and measure the possible welcomeness of a web crawler to websites and further predict the changes of its search engine market share.

\(^1\)http://www.news.com/8301-13578-3-9765451-38.html?tag=replblg
As described above, the favorability network of websites can be studied to find webmaster communities.

Web access logs are a primary source for webmasters to setup their crawlers regulation rules. Thus, the robots.txt files can be also studied as a partial outcome of the access logs of websites. It is typically hard to get the access logs from websites. By studying robots.txt files, we can study spambots such as emailsiphon that extract email addresses from webpages and send spams based on the content analysis of that page. When a webmaster puts the name emailsiphon into the robots.txt files, we can infer that the spambot may periodically access this website.

With the fast growing Web technology, it is very easy to customize a new crawler as well as imitate a well known crawler or browser. It is a non-trivial problem for webmasters to identify crawlers from users. It is important to identify crawler generated logs not simply relying on User-Agent string.

Since the purpose of BotSeer is to provide possible resources for web crawlers related issues, web crawler developers may also need the resources while designing their crawlers.

Based on the above use cases, we design three major components in BotSeer to assist these tasks. We implement the favorability measure \cite{69} on all the robots.txt files we collected. The measure for each website and the statistics over all the websites provide researchers a more efficient tool for related research. A fielded full-text index of robots.txt files provides the necessary functionalities for studying the bias over different crawlers. A large scale log analysis of web access logs provides a valuable resource for identifying crawlers and characterizing them, hence developing more efficient regulations. The search components are based on the open source indexer, Lucene \cite{15}.

Architecture for BotSeer system is illustrated in Figure 5.1.

All the services provided by BotSeer are based on the data obtained from the Web and Web server access logs. A Web crawler handles the data acquisition tasks including getting robots.txt files. The data crawled from the Web is saved in a local repository for further data processing. A data select module handles the data organization and feed the applications that need data inputs. The file parser implements the Robots Exclusion Protocol and parses each rule in robots.txt files to provide fielded searching and filtering for research. The parsed robots.txt files are then indexed using Lucene to provide efficient access to the documents. A fielded index is designed to store the fields extracted according to the Robots Exclusion Protocol as well as the bias analysis result for each robots.txt file. Robot generated Web access logs are analyzed by the log analysis module and the statistical results are stored in MySQL database. A Web based user interface provides the access to each services in BotSeer for users. User queries are handled by a query processor with the support of multiple query modifiers. Three major services, robots.txt search, bias analysis and log analysis, delivers the data and analysis to users.
BotSeer system is planned to crawl 80 million top level domains in the first phase and their subdomains in the second phase. BotSeer updates the robots.txt files in a monthly basis to capture the temporal properties. With the growing size of robots.txt file index, prioritized updating policy is to be applied to BotSeer index. The system is built on many open source development tools including JDK, Apache Tomcat, MySQL, and Lucene which allow a highly scalable system. Current BotSeer system handles complicated search queries and returns results from 2.2 million documents within 1 second. The system is expected to be able to index robots.txt files for 200 million websites (see Figure 5.2).

5.1 DATA

BotSeer provided services are based on robots.txt files and Web server logs. The acquisition, storage and access of each data source is discussed below.

5.1.1 Robots.txt Files

For the use cases discussed previously, robots.txt files are a primary data source for BotSeer to provide the services. It has been reported on the Netcraft web server survey that there are more than 142 million active websites in October 2007. The crawler for BotSeer is required to be able to check a significant portion of all the active websites and find robots.txt files in a short period of time in order to provide up-to-date analyses and information regarding crawler regulation. Traditional Web crawlers rely on parsing Web pages and following hyperlinks to traverse the Web graph. This method usually requires a great deal of resources in order to parse the content.
of Web pages. This is inefficient and unnecessary for acquiring and parsing robots.txt files since valid robots.txt files must always be placed in a standard location relative to the root domain of a website. In addition, relying entirely on links in the Web graph may fail to identify sites that fall in disconnected components within the web graph.

Instead, we use a java based automated agent to collect robots.txt files without explicitly traversing the link structure of the Web. The architecture of the crawler is shown in Figure 5.3. We first collect the active top level domain names from the .com, .net, .org, and .biz domains.
from their respective authoritative root providers, yielding over 80 million distinct hostnames. The robots.txt files are checked under the root directory of these websites and then downloaded directly from each domain where they exist. Our assumption is that most websites from the above domains will be acquired via this method; however, our crawler will not locate and index robots.txt files outside of the root providers. It is an open question whether bots will ever see these files in any case.

A crucial issue for crawling robots.txt files in this manner is that the requests are always pointing to unique domain names. Thus, a single DNS server will not be able to handle the heavy request load (>100 name resolving requests per second beyond a day). We tried several DNS servers within the network service range. All of them stop responding after a few minutes of heavy requests. A DNS request is a major cost in the web crawling process. DNS requests are typically determined by the connection bandwidth between the client and the server. Although a name resolving request over a socket connection can be very fast (several milliseconds), the major cost will be spent in constructing the connection. Multiple DNS servers can be used to feed multithreaded name resolving requests. But these DNS servers will quickly stop responding because of the requirements for crawling speed. Another problem is that some of the available DNS servers do not correctly resolve hosts (e.g. pointing to a specific website or localhost). It is therefore necessary to check the correctness of each DNS server before using it. If all the checks are implemented together in one crawling task, the crawling performance can be significantly slowed. An experiment shows that only 10-20 websites can be checked in one second for this crawling method on a 3.0GHz Xeon PC. Since millions of websites need to be checked, it will take months to check most of the active websites.

In order to address the issues of mass DNS requests, an asynchronous DNS resolving method is implemented into the BotSeer crawler which separates the DNS server checking, the name resolving and the document downloading tasks. A DNS resolving module tests available DNS servers by triangulating multiple known resolving results and simultaneously recording the response time. The correctly available DNS servers are stored in a pool sorted by the response time. A name resolving module reads all the domain names that need to be resolved and sends them to the DNS pool. The best available DNS server will be used to resolve a domain name and then be put into a rest pool until it can be reused. For all the resolved domain names, the robots.txt files can be checked and downloaded by constructing IP-based socket connections to the websites. Such a method provides a lightweight task for each crawling thread and a fast socket connection over IP addresses. We successfully obtain a processing speed result (checking and downloading robots.txt files if available) of 100 websites in one second with this crawling method. We are able to check 13,257,110 websites and download 2,264,820 within a few days on a single Intel 2.8GHz Xeon PC with 1GB RAM (disk access often delays the process in practice). BotSeer also provides a submission system for users to submit robots.txt files directly to the system. The user submission system is integrated with the bias analysis module so that when a user submits a website, it is indexed with the bias analysis result is provided. Since each robots.txt file relates to a unique domain name, the information regarding each domain (whether
the domain has a robots.txt file or not) is also an important resource for the study of crawler regulation. The domain information is stored in MySQL database tables for further analysis.

Since each robots.txt file relates to a unique domain name, the information regarding each domain (whether the domain has a robots.txt file or not) is also an important resource for the study of crawler regulation. The domain information is stored in MySQL database tables with four columns, a unique id for each domain, the domain name, IP address of a domain and the status of the robots.txt file response, stored in separate tables according to the domain suffix. The domain names and IP addresses are indexed for faster access. The .com domains are further split into 27 sub-tables according to the first letter of the domain name for faster access. The robots.txt files are named after the table name and id of each corresponding website.

We also propose to provide access to past robots.txt for longitudinal research. The robots.txt files are stored based on the date that they were downloaded. Since robots.txt files tend to be modified for privacy and security purposes, it is of value to track the changes in robots.txt files to investigate temporal trends. However, storing robots.txt files from each crawl will be redundant since a significant percentage of websites will not show frequent changes in the file. This issue is handled by storing only unique files for each site. A data selection middleware can select a set of robots.txt files according to temporal criteria.

![Diagram: Data selection module between physical storage and applications.]

BotSeer also provides a submission system for users to submit robots.txt files directly to the system. The user submission system is integrated with the bias analysis module so that when a user submits a website, it is indexed with the bias analysis result is provided.

### 5.1.2 Web Server Logs

Web server access logs are one of the most important resources used to study Web crawler behavior since logs have all the access and activity information of generated by crawlers. A typical Web server access log record includes the connection IP address, the requested files, the identity of a crawler, and the time of access. Figure 5.5 shows the distribution of visits of each unique IP address per day. It shows a power-law distribution for the visits less than about 35 connecting with another distribution for large number of visits.

The size of logs for a large Web servers are on the scale of Gigabytes or more for a month. In BotSeer, a 43 Gigabytes CiteSeer\(^2\)[26] Web access log including 86 million entries from May 2005.

\(^2\)http://citeseer.ist.psu.edu
45

Figure 5.5. Distribution of visits per day from each unique IP address.

06, 2006 to Jun. 05, 2006 is used as our data source. A multithreaded log analysis tool was developed to handle the parsing and analysis of large text files with the support of a MySQL database. The log analysis tool is able to analyze and ingest 9 million records in to MySQL database in 30 minutes with 20 threads.

For the study of crawler behavior, the log analysis tool provides statistical data including the total visits, number of requests that disobey robots.txt rules, unique IP addresses with the same User-Agent name, total bandwidth used by crawlers and the overall favorability of a crawler. The statistical data table has five columns, the User-Agent string, the IP address, the amount of visits, the amount of requests disobeying robots.txt and the total size of downloaded files.

5.1.3 Open Source Crawlers

Many software applications rely on customized crawlers to collect information from the Web. Open source crawlers can be seen as a reference for crawler development as well as the resource for webmasters to understand how the web crawlers work so that they can decide which crawlers should be allowed or disallowed. BotSeer indexes 6,514 documentation files and source code of 18 open source crawlers and provides searching and browsing capability over this data.

5.2 WEB APPLICATION

5.2.1 Robots.txt File Search

The goal of BotSeer is to provide information to anyone interested in studying crawler behavior and regulation. The design of the BotSeer web application considers several possible use cases for these users. The most anticipated use case is the search for named crawlers in robots.txt files, and biases associated with specific crawlers. The BotSeer response to the query “botname:msnbot”
Figure 5.6. BotSeer robots.txt search component response to query “botname:msnbot”.

is shown in Figure 5.6. The search results indicate how sites generally treat specific crawlers. Different ranking methods are required for each purpose. For the use case of analyzing the robots.txt files, users are more interested in the robots.txt files from well used website. On the other hand, for the use case of studying how the crawler is received, users are more interested in the websites that favor or disfavor the crawler. The PageRank based ranking method ranks the results by combining the PageRank score [50] of websites and the tf-idf scores [58] between the query term and the robots.txt files. The bias based ranking method ranks the results by the bias of the named crawler presented in robots.txt files. Users are able to switch between ranking methods when they are searching for the User-Agent field for different use cases.

For each returned robots.txt file, BotSeer presents the link to the URL of the robots.txt file, the abstract and cache of the file, and the link to the bias analysis of the file. Users may check the cached robots.txt file as well as link to the original source directly. By clicking the link to bias analysis, users are able to obtain the biases presented in the robots.txt file (the bias analysis is discussed in [69]). The search results give users references on how webmasters regulate the crawler msnbot. The operator of msnbot is also able to see how their crawler is received.

The fielded index provides a wide range of opportunities for BotSeer use cases. Users may specify which field of the robots.txt they are searching for by adding search modifiers including “botname”, “site”, “disallow”, “allow”, “favor”, “disfavor”, “nobias” and “delay”. Each modifier refers to a specific field in the full-text index. “botname” refers to the “User-Agent” field which identifies the name of a crawler. For example the query “botname:msnbot” searches “msnbot” in the User-Agent fields of all robots.txt files. “site” refers to the website names from which
the robots.txt files are collected. "disallow", "allow" and "delay" refer to the corresponding directives in robots.txt files. We also support the modifier "favor", "disfavor" and "nobias" to enable the search of favorability (bias) presented by robots.txt files. With the fielded index, it is possible to study some of the topics we mentioned in section 2 by using BotSeer. For example, a user can search for "favor:googlebot site:gov". BotSeer will return all the government websites that favor googlebot. For the second use case in section 2, user can search for "contain:universal site:gov disallow:/'" to return all the government websites that have the rule of "Disallow:/" and "User-Agent:*" and then find the websites needed with simple further processing. It is also easy to compare the favorability of two or more crawlers. User can search for "favor:googlebot disfavor:msnbot disfavor:slurp" which refers to the websites that favor "Google" but disfavor "MSN" and "Yahoo".

Google.com also indexes robots.txt files. Google users are able to search in the robots.txt files by issuing the query with modifier "inurl:robots.txt filetype:txt". Unlike the BotSeer search component which parses robots.txt files according to the Robots Exclusion Protocol, Google search indexes robots.txt files as normal plain text files. A comparison of Google search results for "msnbot" and BotSeer results shows that BotSeer provides more information on the crawlers from a regulation perspective. (On BotSeer homepage, users can click on the Google button to perform a robots.txt file search in Google.)

5.2.2 Crawler Search

The crawler behavior search component searches the bias analysis results and crawler generated logs from a large Web server. The log is generated by 8,932 crawlers from 61,204 IP addresses. This component is designed for users to check the real-world behavior of a named crawler. Users may submit a crawler name as the query to the system. The system matches the query to the bias analysis records and log records generated by corresponding crawlers and returns statistical data for the crawlers.

The result page lists the records in the log with bias toward a crawler, User-Agent strings contain the name of the crawler, and the geographical distribution of the crawler’s IP address (see Figure 5.7). When users click on the User-Agent string, BotSeer will present the detailed information about this User-Agent string including the type of the crawler, the actual IP of the crawler, and the registered name of the IP addresses. If the registered name of IP addresses does not match the name of the crawler, user will know this is a potential crawler using fake names.

Since the User-Agent field in the log is specified by the crawlers in the HTTP request header, it is possible for any crawler to “pretend” to be and use the name of another crawler. But the IP addresses where the crawlers are actually from are harder to spoof. It is necessary to check the IP addresses as well as the name specified in the User-Agent string to identify a crawler. From the example shown in Figure 5.7, part of the log records are generated by the crawlers who named themselves as Googlebot while we believe they are not. By clicking specific User-Agent strings in the crawler search result page, BotSeer presents the detailed information about the
User-Agent string (see Figure 5.8). Users can click on the IP addresses listed on the result page and BotSeer will show users the WHOIS information\(^3\) about that IP address. For the example of “Googlebot” records, the IP addresses that obeyed the Robots Exclusion Protocol in fact those allocated to Google Inc. We also plot the geographical locations of each visiting crawler. As an example, figure 5.9 shows the geographical distribution of crawlers that visit CiteSeer. Each gray point represents a unique crawler IP address. We use the blue color to circle out “Googlebots” that obey the robots.txt of CiteSeer and red color to circle out fake “Googlebots” that disobey the robots.txt.

We also design a honeypot, a set of sites where each site is configured with a distinct regulation specification using the crawlers exclusion protocol in order to capture specific behaviors of web crawlers. A detailed log analysis on the test websites will reveal how well a web crawler behaves. The analysis of honeypot logs will provide rules to measure the ethicality of web crawlers. The analysis of web access logs is a very important part of BotSeer not only for searching crawler behavior but also for possible new regulation standards. Much data could be generated by

\(^3\)Whois information is provided by GeekTool.com.
combining the bias analysis results and the log analysis results to study the true behavior of web crawlers. With the log analysis results, BotSeer can also be used as a reference to identify poorly behaved crawlers and crawlers pretending to be browsers and other well known crawlers. Therefore, BotSeer can be a service to the webmaster community.

5.2.3 Data Analysis

BotSeer provides two types of statistical data analysis. The bias analysis module analyzes the bias presented in the robots.txt files, indicating how specific crawlers are regulated on the Web. Bias metrics can also be used to evaluate the reputation of specific crawlers. The crawler-generated log analysis provides the statistics of the crawlers’ actual behavior on a Web server.
5.2.3.1 Bias Analysis

Websites can explicitly specify an access preference for each crawler by name. Such biases can be used as a resource to study the regulation of Web crawlers. The bias analysis module analyzes the biases in robots.txt files based on the bias measure [69]. A favored crawler is defined as a crawler allowed to access more directories than the universal crawler according to the robots.txt file in the website. The universal crawler ‘*’ is any crawler that has not matched any of the specific user-agent names in the robots.txt file. In other words, the universal crawler represents all the crawlers that do not appear by name in the robots.txt file. The detailed discussion about the bias analysis algorithms and results can be found in [69]. BotSeer implements the bias definition and provides the bias analysis for each robots.txt files. Users are able to get the bias analysis result by following the hyperlink of “Analyze it!” on the search result page of robots.txt files. The hyperlink will lead users to the robots.txt analysis result page (see Figure 5.10).

![Robots.txt Test and Analysis](image)

**Figure 5.10.** Detailed bias analysis of a website.

BotSeer system also analyzes the overall bias on a set of robots.txt files based on the favorability defined in [69]. An example of a favorability analysis on 1,858 web crawlers is shown in Figure 5.11.

Users can click on each column title, and the result can be ranked by the times the crawler’s name appeared, the times it was favored, times disfavored, and favorability respectively. The hyperlink of the crawler name will direct users to the log search page.
5.2.3.2 Dynamic Bias Analysis

To better serve the study of crawler bias in a limited set of websites, the BotSeer system provides a dynamic bias analysis tool which provides the bias analysis over a website list generated by Google search engines. Users can search for a specific topic of interest by typing in a keyword phrase. The dynamic bias analysis module will submit the keyword phrase to Google and generate a list of websites from the returned results. The tool will check the robots.txt file for each website in the list, generating an analysis dynamically regarding the bias metrics for each crawler regulated in the collected set of robots.txt files. The results are expected to reflect the bias of a top-ranked websites for a given topic.

The dynamic bias analysis module (DBAM) is a four step process. The first step handles the general query results by accessing the public APIs provided by Google. The prototype of BotSeer limits the analysis to the top 10 results for faster service. The second step passes the results set to the data selection module introduced previously and generates a virtual directory that contains the robots.txt files of the result sites. Since our data is stored according to the time of the crawl, the data selection module is also able to generate virtual directories for specific dates. Thus, the dynamic bias analysis module can also provide historical analysis on biases of crawlers. The third step is to analyze the biases in the selected robots.txt files. The bias analysis component will analyze the selected robots.txt files as we described above. The last step is to present the bias results for different search engines to users.

The dynamic bias analysis module is designed to aid the study of the biases toward crawlers in various topics of websites. For example, users might be interested in the bias analysis on newspaper websites. They can input a query such as “robots exclusion protocol” to generate
a collection of websites. Users can even input a customized list of websites to the module and analyze the biases (see Figure 5.12). The total response time with cached robots.txt files is less than 2 seconds (Google API search takes about 0.7 seconds to return results).

![BotSeer](image)

**Figure 5.12.** Dynamic Bias analysis on the query “Robots Exclusion Protocol”.

### 5.2.3.3 Robot Generated Log Analysis

The crawler-generated log analysis component parses the Web log file from multiple websites and indexes the user-agent strings and corresponding IP addresses and behaviors. The crawler behavior statistics are extracted from the Web log including number of visits, number of downloads and number of visits disobeying the Robots Exclusion Protocol. The result page of the statistics of crawler generated visits of CiteSeer is shown in Figure 5.13. Users can sort the result by each column.

The visits distribution and geographical distribution of crawler-generated logs can also be monitored through the log analysis system (see Figure 5.15).

The crawler generated visits are also compared to user generated visits (see Figure ??).
5.3 DISCUSSION

There is much information contained in robots.txt files. Robots.txt files reflect the bias of the websites toward specific web crawlers, and also show the behavior of web crawlers visiting the websites. The study of crawler bias [70] shows that most of the crawler names appeared in the robots.txt files only once or twice which suggests that which crawler names appeared in the robots.txt files are not decided based on global acknowledgements. The names may come from a community that discuss the well behaved and bad behaved crawlers or a website development company with a standard crawler policy. In this case, robots.txt files reflects new relationships among the websites based on the community or developer. The design of robots.txt rule is also a tradeoff between attracting more traffic and not wasting bandwidth to non-user access. In such case, robots.txt files reflects the website development strategy.

BotSeer system is designed to provide automated data analysis on the entire World Wide Web robots.txt file collection and large scale Web access logs. The architecture of BotSeer is scalable and can collect millions of robots.txt files and process very large Web access logs. The system can be used for various design and analysis uses, some of which we now list. The robots.txt file search can be used to study how a crawler is regulated on the Web. Users can search for
“cgi-bin” in the “disallow” field to see how many websites disallow this directory. Users can also search for a special term or phrase used in a robots.txt file to check how many websites are using the same robots.txt file. (There are many cases in our database that the robots.txt files on two or more websites are exactly the same.)

The design of the system relies on well supported open source software which is scalable and capable of handling large databases. We successfully implemented BotSeer system with 4.6 million robots.txt files out of 17 million websites. The full text searching service return an average query in less than 1 second.

Figure 5.14. The crawler generated traffic analysis and monitor.
Figure 5.15. Comparison of user visits and crawler visits. The press release shows two dates that BotSeer is reported in news media.
Crawler Behavior Analysis

Since the REP serves only as an unenforced advisory to crawlers, web crawlers may ignore the rules and access part of the forbidden information on websites. Therefore, the usage of the Robots Exclusion Protocol and the behavior of web crawlers with respect to the robots.txt rules provide a foundation for a quantitative measure of crawler ethics.

It is difficult to interpret ethicality in different websites. The unethical actions for one website may not be considered unethical in others. We follow the concept of crawler ethics discussed in [23, 73] and define the ethicality as the level of conformance of crawlers activities to the Robots Exclusion Protocol. In this thesis, we propose a vector model in the REP rule space to formulate and measure the ethicality of web crawlers computationally.

6.1 Models

This section describes the modeling of crawler behavior and ethical factors in our research. The notions and definitions are described in the following subsections.

6.1.1 Vector Model of Crawler Behavior

In our research, each web crawler’s behavior is modeled as a vector in the rule space where rules are specified by the Robots Exclusion Protocol to regulate the crawler behavior. The notions of basic concepts related to ethicality are introduced as the following:

- A rule is a dimension in the rule space. Each rule is defined as an item from a complete set of rules that describe all the regulations of web crawlers.

- For a given subset of N rules $R = \{r_1, r_2, ..., r_N\}$, a web crawler’s behavior is defined as an N-vector $C = \{c_1, c_2, ..., c_N\}$ such that $c_i > 0$ if the crawler disobey the rule $i$ and $c_i = 0$ otherwise.
• For a given subset of $N$ rules $R = \{r_1, r_2, ..., r_N\}$, ethical weight is defined as an $N$-vector $W = \{w_1, w_2, ..., w_N\}$ in the rule space where $w_i$ is the cost for disobeying rule $i$.

• For a given rule $r$, $N_v(C)$ is defined as the number of visits generated by crawler $C$ that violate or misinterpret rule $r$, and $N(C)$ is defined as the total number of visits generated by crawler $C$. $P(C|r) = \frac{N_v(C)}{N(C)}$ is the conditional probability that crawler $C$ violates rule $r$.

### 6.1.2 Ethicality Metrics

The statistics of web access logs show that there are significant problems in crawler ethics. As discussed in [73], there can be consequences, including denial of service, cost and privacy, if web crawlers do not crawl ethically. From a technical perspective, denial of service and cost refer to the same crawler behavior - generating unnecessary traffic to a website. Privacy refers to the behavior of crawlers accessing restricted content of a website. Web crawler generated ethical issues can be interpreted differently with different philosophies of determining ethicality. We introduce four different models of ethicality measures to show how ethicality can be evaluated differently.

#### 6.1.2.1 Binary Model

In certain cases, one unethical act makes a crawler unethical. The binary model is based on this strong assumption and reflects whether a crawler has ever violated any rules (see Eq6.1).

$$E_{bin}(C) = 1 \left( \sum_{r \in R} (N_v(C)) > 0 \right)$$ (6.1)

where $N_v(C)$ is the number of visits generated by crawler $C$ violating or misinterpreting robots.txt files, $1(x)$ is the indicator function where $1(x) = 1$ when crawler visits have violations or misinterpretations, and $1(x) = 0$ otherwise. Because of the strong assumption in the binary model, only clearly described rules without ambiguity in REP should be considered. For the binary model, a crawler is ethical only if it obeys all rules at all times. Any exception classifies the crawler as unethical.

#### 6.1.2.2 Probabilistic Model

The goal of measuring ethicality may be to predict possible behaviors of web crawlers. The probabilistic ethicality model measures the likelihood of a crawler violating or misinterpreting the rules specified by a website. Probabilistic ethicality is defined as the probability of a crawler accessing restricted content inside a website (see Eq6.2).

$$E_{prob}(C) = P(C) = \sum_{r \in R} (P_{usage}(r) \times P(C|r))$$ (6.2)
where $P_{usage}(r)$ is the probability of rule $r$ being specified in a website and $P(C|r)$ is the probability of crawler $C$ violating or misinterpreting rule $r$. The joint probability $P_{usage}(r) \times P(C|r)$ represents the likelihood of crawler $C$ misinterpreting rule $r$. The larger the probability, the more likely a crawler will violate or misinterpret robots.txt files.

### 6.1.2.3 Relative Model

Measuring the ethicality of a crawler is sometimes meaningful when comparing to other crawlers. The relative model measures the ethicality of web crawlers relative to all other crawlers. Relative ethicality can be defined as the probability of rule $r$ being violated or misinterpreted by crawler $C$ weighted by the percentage of crawlers violating or misinterpreting the rule (see Eq 6.3).

$$E_{relative}(C) = \sum_{r \in R} ((1 - P_{violate}(r)) \times P(C|r))$$

(6.3)

where $P_{violate}(r)$ is the percentage of crawlers violating or misinterpreting rule $r$ and $P(C|r)$ is the probability of crawler $C$ violating or misinterpreting rule $r$. A crawler’s ethicality is penalized for violating rules that are obeyed by other crawlers. The higher the score, the more likely the crawler is to violate more rules than other crawlers.

### 6.1.2.4 Cost Model

More generally, an ethical cost vector can be defined in the rule space. Each element of the cost vector represents the cost of violating the corresponding rule. The ethicality $E_{cost}(C)$ of crawler $C$ is then defined as the inner product of the crawler vector and the ethical cost vector (see Eq 6.4).

$$E_{cost}(C) = \langle C \cdot W \rangle = \sum_i c_i \times w_i.$$  

(6.4)

The ethical weight vector can be defined based on the actual costs of disobeying regulation rules in different circumstances. The elements of an ethical weight vector do not necessarily constitute a fixed value. Instead, cost functions can be applied to measure ethicality for different crawlers with better granularity. For small websites with limited internet bandwidth, crawlers that do not set a proper visiting frequency or that repeatedly crawl the same content ignoring its recentness may be considered unethical. Crawlers failing to update the restriction policy and resulting in maintenance and presentation of outdated content to the public may be considered unethical in e-commerce websites. Crawlers trying to access restricted information or information protected with authentication designed for use in a limited group may also be considered unethical for many websites. Because of the differences in the content and functionality of websites, it is hard to measure ethicality as a universal factor.

We propose two quantitative cost functions, namely content ethicality $E_c$ and access ethicality $E_a$ to evaluate the ethicality of web crawlers from a general regulation perspective. In
content ethicality, cost is defined as the number of restricted webpages or web directories being unethically accessed. More specifically, for a set of websites \( W = \{ w_1, w_2, ..., w_n \} \), \( D(w_i) \) is the set of directories restricted in its robots.txt file of \( w_i \). \( V_C(w_i) \) is a subset of \( D(w_i) \), which is visited by crawler \( C \) by violating or misinterpreting the rules in the robots.txt file. Content ethicality is defined in Equation 6.5.

\[
E_c(C) = \sum_{w_i \in W} \frac{|V_C(w_i)|}{|D(w_i)|}.
\] (6.5)

The definition of content ethicality is intuitive. The more rules a crawler is violating, the higher the content ethicality score it will receive. According to the definition, the content ethicality score is a real number between 0 and 1.

Access ethicality can be defined as the visit interval of a crawler to a website with respect to the desired visit interval of the website. The desired visit interval for a website can be obtained from the crawl-delay rule in its robots.txt file. Since web crawlers are automated programs that traverse the web, the visit interval for each website depends on the crawling policy of crawlers. With access ethicality, we assume the visit interval for each crawler is proportional to the incoming links of a website (inlinks for each website can be estimated by the link results from Google). Thus, we can estimate the visit interval of each crawler for all websites by observing the visit interval for one website. For a set of sample websites \( W = w_1, w_2, ..., w_n \), the visit interval \( interval_C(w_i) \) of crawler \( C \) for site \( w_i \) can be estimated by the visit interval \( interval_C(w_a) \) for site \( w_a \), \( interval_C(w_i) = \frac{\text{inlink}(w_a)}{\text{inlink}(w_i)} \times interval_C(w_a) \). If a crawler obeys the crawl-delay rule, the lower bound of the interval is determined by the crawl-delay rules specified in robots.txt files. Access ethicality can be defined as Equation 6.6.

\[
E_a(r) = \sum_{w_i \in W} \frac{e^{-(interval_C(w_i)-\text{delay}(w_i))}}{1+e^{-(interval_C(w_i)-\text{delay}(w_i))}}
\] (6.6)

For websites without crawl-delay entries, the default delay value is set to 0 since there is no desired crawl interval. According to the definition, the access ethicality score is normalized from 0 to 1, with lower scores representing crawlers that respect the desired crawling interval. When the crawlers obey the crawl-delay rule, access ethicality scores are less than 1/2.

Based on the definition of content ethicality and access ethicality, the following algorithms can be used to compute content ethicality and access ethicality scores of web crawlers. The model assumes that the behavior of web crawlers does not vary significantly across websites. This means that if a crawler violates a rule in one website, we assume it will violate similar rules in other websites. The algorithms take the ethicality vector as input. The violated directory set is obtained by evaluating each restricted directory against the violation table of crawlers. Thus, for a given robots.txt file \( f \) and web crawler \( C \), the algorithms return the set of directories that can be unethically accessed by crawler \( C \) (see Algorithm 2). For all the websites with robots.txt files in our collection, the ethicality score is computed by normalizing the ethical access for each
crawler in each website according to our definition of content ethicality.

Algorithm 2 GetContentEthicality(f,C)

1: \( D \) all directories in \( f \);
2: if \( r \) is in \( f \) then
3: \( D_C = \) directories disallowed for \( C \)
4: else
5: \( D_C = \) directories disallowed for *
6: end if
7: for \( d \in D_C \) do
8: if \( C \) can access \( d \) then
9: \( V_C \leftarrow d \)
10: end if
11: end for
12: RETURN \( V_C, D \)

The computation of access ethicality scores is described in Algorithm 3. The overall access ethicality score for each crawler is obtained by averaging the access ethicality scores of all websites in our collection.

Algorithm 3 GetAccessEthicality(f,C)

1: get inlink(\( f \)) from Google
2: interval\(_C(\( f \)) = \frac{\text{inlink}(k)}{\text{inlink}(\( f \))} \cdot \text{interval}_r(k)\)
3: if \( r \) is in \( f \) then
4: delay(\( f \)) = Crawl-Delay rule for \( C \)
5: else
6: delay(\( f \)) = Crawl-Delay rule for *
7: end if
8: if \( \text{interval}_C(\( f \)) < \text{delay} \) AND \( C \) obeys Crawl-Delay rule then
9: \( A_C = \frac{1}{2} \)
10: else
11: \( A_C = \frac{1}{1+e^{-(\text{interval}_C(w_j)-\text{delay}(w_j))}} \)
12: end if
13: RETURN \( A_C \)

6.2 Experiments

6.2.1 Crawler Behavior Test: Honeypot

The ideal way to get ethicality scores for each web crawler would be to collect the access logs of all websites. This is impossible, however, since websites typically do not share their access logs for a variety of reasons. To address the data collection issue, we set up a honeypot that records crawler behavior under different circumstances. The behavior traces collected from the honeypot can be used to deduce the ethicality of all web crawlers visiting the honeypot and to estimate crawler ethicality at large.
The honeypot is designed based on the specifications in the Robots Exclusion Protocol and common rule cases derived from our prior study of 2.2 million sample websites with robots.txt files, including all cases where the robots.txt rules can be violated by crawlers. Table 6.1 shows the usage of rules in our robots.txt collection. Each website in the honeypot tests one or more specific cases against each crawler.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contain disallow rules</td>
<td>913,330</td>
</tr>
<tr>
<td>Contain conflicts</td>
<td>465</td>
</tr>
<tr>
<td>Contain octet directives</td>
<td>500</td>
</tr>
<tr>
<td>Use variations of user-agent name</td>
<td>45,453</td>
</tr>
<tr>
<td>Use multiple user-agent names in one line</td>
<td>42,172</td>
</tr>
<tr>
<td>Use multiple directories in one line</td>
<td>3,464</td>
</tr>
<tr>
<td>Use crawl-delay rule</td>
<td>39,791</td>
</tr>
<tr>
<td>Use visit-time rule</td>
<td>203</td>
</tr>
<tr>
<td>Use request-rate rule</td>
<td>361</td>
</tr>
</tbody>
</table>


**Honeypot 1** This website tests crawlers’ interpretation of the *Disallow* rule. There are two hyperlinks on the root page of the honeypot 1, /d1/ and /d1/d01/. The robots.txt file specifies two rules, *Disallow : /d1/ and Allow : /d1/d01/.* The rules should be interpreted as meaning that all files under directory /d1/ including /d1/d01/ are restricted based on the REP, although the second rule conflicts the first one. The REP specifies that “The first match found is used.” Thus, the conflict should be resolved by following the first rule. (To allow /d1/d01/ and disallow other directories in /d1/, the correct setting should be to list the allow rules prior to the disallow rules.) In honeypot 1, if a crawler visits /d1/, it does not obey the *Disallow* rule. If a crawler visits /d1/d01/ but not /d1/, it does not resolve the subdirectory conflict correctly.

![Figure 6.1](image.png)

**Honeypot 2** This website tests crawler behavior when the rules directly conflict with each other. In this case, a hyperlink points to a page under directory /d1/. The robots.txt file includes *Disallow : /d1/ and Allow : /d1/ in the given order.* There is obviously a conflict between the two rules. If the crawler visits pages under /d1/, it does not resolve the conflict.

**Honeypot 3** This website has the same page structure as honeypot 1 but a different robots.txt file. There is only one rule, *Disallow : /d1/.* REP requires crawlers to exclude any URLs starting with /d1/. Thus, /d1/d01/ being restricted is implied by the robots.txt file. If a crawler visits /d1/d01/, it fails to parse the rule correctly.
Honeypot 4 This website tests the robots.txt update policy of crawlers. The rules in robots.txt change after a period of time. Crawlers should update the robots.txt files periodically to respect the latest access regulation. If a crawler does not update robots.txt files as needed, restricted content may be accessed unethically. There are two pages /d1/ and /d2/ linked by the homepage of the website. /d1/ is restricted in time period 1 and allowed in time period 2. /d2/ is allowed in time period 1 and disallowed in time period 2. The timestamps of crawler requests to these two pages show the crawlers’ update policy. This site provides evidence to study how search engines deal with web pages that change permissions over time.

Honeypot 5 This website tests how crawlers deal with errors in robots.txt. A few error lines are placed before the rule Disallow : /d4/ in the robots.txt file. Thus, if a crawler obeys disallow rules in other honeypots but still visits web pages under /d4/, it is considered to have failed in handling the errors. It is not the crawlers’ responsibility to correct the errors. However, ethical robots should at least ignore the errors and still obey the correctly formatted rules.

Honeypot 6 This website tests how crawlers match the User – Agent field. The robots.txt file in this site is dynamically generated by a server side script that parses the User-Agent string from the HTTP request header and generates substrings. The Robots Exclusion Protocol specifies that the User – Agent field is case insensitive and that a crawler should match a token
that describes it. Substring and variations of crawler names are often used in the robots.txt files to specify the regulation rules. For example, `google` as a crawler name appeared in 16,134 robots.txt files in our collection. Crawlers may ignore the name, however, because it is not the exact name. There are six directories linked by the homepage of this website. Each directory is restricted to one substring of the incoming crawler except one open directory. In such settings, if a crawler requests the open directory, the one directory the crawler does not request shows the matching mechanism of the crawler. If all directories are requested by a crawler, it is considered unethical.

**Honeypot 7** This website tests the rule for status code regulation. A request to the robots.txt file in this website will receive a status code response of 403. According to REP, when the request to robots.txt receives an HTTP status code of 403 or 401, “a robot should regard access to the site completely restricted.” If a crawler ignores the rule and visits any pages in the website, it is considered unethical.

**Honeypot 8** This website tests whether a crawler is a spam crawler. In this website, there
is no explicit link to the /email/ directory but /email/ is restricted in the robots.txt file. Thus, spambots can be identified if they try to explore the hidden directory /email/.

```
User-Agent: *
Disallow: /email/
```

Figure 6.8. Site structure and robots.txt file for Honeypot 8.

**Honeypot 9** This website tests octet conversion related rules in the REP. If a %xx encoded octet is encountered, it should be un-encoded prior to comparison unless it is %2f. The page /a−d.html is linked by the homepage of the website, and the robots.txt file disallows /a%2dd.html. Thus, crawlers should not crawl the file /a−d.html.

```
User-Agent: *
Disallow: /tmp
Disallow: /temp/
Disallow:/a%2dd.html
Disallow:/a%2fd.html
Disallow:/%7et2/index.html
```

Figure 6.9. Site structure and robots.txt file for Honeypot 9.

**Honeypot 10** This website tests whether a crawler respects META-Tag rules specified in a webpage. The rule < META NAME = "ROBOTS" CONTENT = "NOINDEX, NOFOLLOW" > is specified in a webpage. According to the rule, any hyperlinks on this page should not be followed, and corresponding webpages should not be requested.

**Honeypot 11** This website tests whether a crawler respects the desired crawl delay. The robots.txt of this site sets the delay time to 200 seconds and then to 18,000 seconds after two months. The interval between consecutive visits from one crawler is compared to the desired delay time. Access ethicality can be computed based on crawler behavior from this site.

Our settings of honeypot websites only test currently available crawler regulation rules.

### 6.2.2 Results

We monitor the web access logs from within the honeypot and generate statistics based on the crawlers that visit the honeypot. The probability of violating or misinterpreting each rule specified in our honeypot is listed in Table 6.2.
Table 6.2. Probability of a rule being violated or misinterpreted.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability of being violated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conflict Subdir</td>
<td>4.6%</td>
</tr>
<tr>
<td>Conflict Dir</td>
<td>5.569%</td>
</tr>
<tr>
<td>Disallow Dir</td>
<td>0.969%</td>
</tr>
<tr>
<td>Timely Update Regulation</td>
<td>1.937%</td>
</tr>
<tr>
<td>Ignore Errors</td>
<td>0.726%</td>
</tr>
<tr>
<td>Crawl-Delay</td>
<td>6.78%</td>
</tr>
<tr>
<td>Substring of crawler name</td>
<td>1.695%</td>
</tr>
<tr>
<td>401/403 restriction</td>
<td>9.443%</td>
</tr>
<tr>
<td>Hidden Dir</td>
<td>0</td>
</tr>
<tr>
<td>Meta Tag</td>
<td>0.242%</td>
</tr>
<tr>
<td>Octec Conversion</td>
<td>0.968%</td>
</tr>
<tr>
<td>Case Sensitivity</td>
<td>3.39%</td>
</tr>
</tbody>
</table>

The 401/403 restriction and the crawl-delay rules have a high probability of being violated. Since the crawl-delay rule is an extension to the REP and the 401/403 restriction rule is not explicitly specified in robots.txt files, the violation results are expected. The results also suggest, however, the importance of making REP official and complete. There is also a high probability for crawlers misinterpreting conflicting rules, which demonstrates a need for robots.txt file authors to design and check their rules more carefully.

6.2.2.1 Binary Ethicality

The robot access logs for all the websites in our honeypot are analyzed to extract the binary ethicality vector of each crawler. The ethicality vector is constructed with each row representing a specific crawler and each column representing a specific rule (see Table 6.3). Since not all crawlers visit every sub-site of our honeypot, crawlers should not be compared directly in terms of the number of rules they violated or misinterpreted. For example, Googlebot violates or misinterprets 7 rules in the rule space and HyperEstraier violates 6. However, HyperEstraier violates every rule it encounters. The binary ethicality vector shows that MSNbot is more ethical than Yahoo Slurp and that Yahoo Slurp is more ethical than Googlebot. The results also show that the interpretation of robots.txt rules for each crawler varies significantly. The binary violation table reduces the violation to either 0 or 1. For some applications, a full violation table with the number of total violations for each rule is more useful (see Table 6.4).

6.2.2.2 Probabilistic Ethicality

Binary ethicality does not capture differences among rules. Violations of Disallow rules are obviously less ethical than the misinterpretation of a subdirectory conflict. According to the definition of probabilistic ethicality, each rule is weighted by the probability of being used in a website. The top 15 unethical crawlers ranked by probabilistic ethicality are listed in Table 6.5.

---

1 We only list 18 crawlers of interest. Ethicality of other crawlers that visited our honeypot can be searched through BotSeer http://botseer.ist.psu.edu.
Table 6.3. The binary violation table of web crawlers derived from honeypot access logs.
<table>
<thead>
<tr>
<th>User-Agent</th>
<th>Conflict</th>
<th>Conflict</th>
<th>Disallow</th>
<th>Timely</th>
<th>Ignore</th>
<th>Crawl-</th>
<th>Substring</th>
<th>401/403</th>
<th>Hidden</th>
<th>Meta</th>
<th>Otec</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Subdir</td>
<td>Dir</td>
<td>Dir</td>
<td>Update</td>
<td>Errors</td>
<td>Delay</td>
<td>of crawler name</td>
<td>restriction</td>
<td>Dir</td>
<td>Tag</td>
<td>Conv</td>
<td>Sensitivity</td>
</tr>
<tr>
<td>Googlebot/2.1</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Googlebot-image</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>hyperestraier/1.4.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>yahoo! slurp</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>msnbot/1.0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>msnbot-media/1.0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>nutch-0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>gigabot/3.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heritrix/1.12.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>panscient.com</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>charlotte/1.0b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>visbot/2.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>arietis/nutch-0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Teemer Netseer</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ask jeeves/teoma</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>netcraft web server survey</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ia_archiver</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>botseer/1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.4. The full violation table of web crawlers derived from honeypot access logs.
Table 6.5. Probability Ethicality of web crawlers.

<table>
<thead>
<tr>
<th>Crawler</th>
<th>Probabilistic Ethicality</th>
</tr>
</thead>
<tbody>
<tr>
<td>hyperestraier/1.4.9</td>
<td>0.0444</td>
</tr>
<tr>
<td>teemer</td>
<td>0.0161</td>
</tr>
<tr>
<td>heritrix/1.12.1</td>
<td>0.0132</td>
</tr>
<tr>
<td>msnbot-media/1.0</td>
<td>0.0054</td>
</tr>
<tr>
<td>googlebot-image/1.0</td>
<td>0.0025</td>
</tr>
<tr>
<td>gigabot/3.0</td>
<td>0.0021</td>
</tr>
<tr>
<td>Yahoo! Slurp</td>
<td>0.0016</td>
</tr>
<tr>
<td>googlebot/2.1</td>
<td>0.0014</td>
</tr>
<tr>
<td>panscient.com</td>
<td>0.0004</td>
</tr>
<tr>
<td>Yahoo! Slurp/3.0</td>
<td>0.0002</td>
</tr>
<tr>
<td>arietis/mutch-0.9</td>
<td>0.0001</td>
</tr>
<tr>
<td>charlotte/1.0b</td>
<td>0.0001</td>
</tr>
<tr>
<td>visbot/2.0</td>
<td>0.0001</td>
</tr>
<tr>
<td>msnbot/1.1</td>
<td>0.0001</td>
</tr>
<tr>
<td>msnbot/1.0</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

The probabilistic ethicality results show that most crawlers are very unlikely to violate or misinterpret the REP rules.

6.2.2.3 Relative Ethicality

Relative ethicality compares web crawlers with each other. The top 15 unethical crawlers measured by relative ethicality are listed in Table 6.6.

Table 6.6. Relative ethicality of web crawlers.

<table>
<thead>
<tr>
<th>Crawler</th>
<th>Relative Ethicality</th>
</tr>
</thead>
<tbody>
<tr>
<td>hyperestraier/1.4.9</td>
<td>0.1288</td>
</tr>
<tr>
<td>panscient.com</td>
<td>0.1137</td>
</tr>
<tr>
<td>Yahoo! Slurp/3.0</td>
<td>0.0835</td>
</tr>
<tr>
<td>googlebot-image/1.0</td>
<td>0.0651</td>
</tr>
<tr>
<td>googlebot/2.1</td>
<td>0.0646</td>
</tr>
<tr>
<td>charlotte/1.0b</td>
<td>0.0613</td>
</tr>
<tr>
<td>heritrix/1.12.1</td>
<td>0.049</td>
</tr>
<tr>
<td>Yahoo! Slurp</td>
<td>0.049</td>
</tr>
<tr>
<td>msnbot/1.0</td>
<td>0.0393</td>
</tr>
<tr>
<td>msnbot/1.1</td>
<td>0.0374</td>
</tr>
<tr>
<td>ask jeeves/teoma</td>
<td>0.0359</td>
</tr>
<tr>
<td>woriobot</td>
<td>0.0359</td>
</tr>
<tr>
<td>arietis/mutch-0.9</td>
<td>0.0342</td>
</tr>
<tr>
<td>gigabot/3.0</td>
<td>0.0328</td>
</tr>
<tr>
<td>teemer</td>
<td>0.0316</td>
</tr>
</tbody>
</table>

The relative ethicality results are defined to compare the correctness of interpreting REP rules among crawlers. The scores can be interpreted as how badly a crawler interprets the REP rules. The results show that all crawlers except hyperestraier and panscient.com are fairly good
at interpreting REP rules.

6.2.2.4 Cost Ethicality

Table 6.8 lists content and access ethicality results for top crawlers that visited our honeypot during the time of the study.

<table>
<thead>
<tr>
<th>User-agent</th>
<th>Content Ethicality</th>
</tr>
</thead>
<tbody>
<tr>
<td>hyperestraier/1.4.9</td>
<td>0.95621</td>
</tr>
<tr>
<td>Teemer</td>
<td>0.01942</td>
</tr>
<tr>
<td>msnbot-media/1.0</td>
<td>0.00632</td>
</tr>
<tr>
<td>Yahoo! Slurp</td>
<td>0.00417</td>
</tr>
<tr>
<td>charlotte/1.0b</td>
<td>0.00394</td>
</tr>
<tr>
<td>gigabot/3.0</td>
<td>0.00370</td>
</tr>
<tr>
<td>nutch test/nutch-0.9</td>
<td>0.00316</td>
</tr>
<tr>
<td>googlebot-image/1.0</td>
<td>0.00315</td>
</tr>
<tr>
<td>Ask Jeeves/Teoma</td>
<td>0.00302</td>
</tr>
<tr>
<td>heritrix/1.12.1</td>
<td>0.0029</td>
</tr>
<tr>
<td>googlebot/2.1</td>
<td>0.00282</td>
</tr>
<tr>
<td>woriobot</td>
<td>0.00246</td>
</tr>
<tr>
<td>panscient.com</td>
<td>0.00202</td>
</tr>
<tr>
<td>Yahoo! Slurp/3.0</td>
<td>0.00136</td>
</tr>
<tr>
<td>msnbot/1.0</td>
<td>0.00049</td>
</tr>
</tbody>
</table>

Table 6.7. Content ethicality scores for crawlers that visited the honeypot.

<table>
<thead>
<tr>
<th>User-agent</th>
<th>Access Ethicality</th>
</tr>
</thead>
<tbody>
<tr>
<td>msnbot-media/1.0</td>
<td>0.3317</td>
</tr>
<tr>
<td>hyperestraier/1.4.9</td>
<td>0.3278</td>
</tr>
<tr>
<td>Yahoo! Slurp/3.0</td>
<td>0.2949</td>
</tr>
<tr>
<td>Yahoo! Slurp</td>
<td>0.2949</td>
</tr>
<tr>
<td>Teemer</td>
<td>0.2744</td>
</tr>
<tr>
<td>Arietis/Nutch-0.9</td>
<td>0.0984</td>
</tr>
<tr>
<td>msnbot/1.0</td>
<td>0.098</td>
</tr>
<tr>
<td>disco/Nutch-1.0-dev</td>
<td>0.0776</td>
</tr>
<tr>
<td>ia_archiver</td>
<td>0.077</td>
</tr>
<tr>
<td>ia_archiver-web.archive.org</td>
<td>0.077</td>
</tr>
<tr>
<td>gigabot/3.0</td>
<td>0.0079</td>
</tr>
<tr>
<td>googlebot/2.1</td>
<td>0.0075</td>
</tr>
<tr>
<td>googlebot-image/1.0</td>
<td>0.0075</td>
</tr>
</tbody>
</table>

Table 6.8. Access ethicality scores for crawlers that visited the honeypot.

Cost ethicality ranks the crawlers differently than probabilistic ethicality and relative ethicality because the cost ethicality scores consider the actual deployment of the robots.txt rules. For example, Googlebot ranked lower in cost ethicality score because it is more favored than other crawlers so that there are fewer cases where Googlebot can violate the rules.

The cost ethicality measure not only considers the penalty for not strictly obeying the rules in robots.txt files but also grants additional ethical points to crawlers that exceed regulations
Table 6.9. Additional ethical behavior of crawlers.

<table>
<thead>
<tr>
<th>User-agent</th>
<th>Parsing Errors</th>
<th>Update policy</th>
<th>Variation of Botname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Googlebot/2.1</td>
<td>yes</td>
<td>remove from index</td>
<td>no</td>
</tr>
<tr>
<td>ia_archiver</td>
<td>yes</td>
<td>n/a</td>
<td>yes</td>
</tr>
<tr>
<td>msnbot/1.0</td>
<td>yes</td>
<td>remove cache (still searchable)</td>
<td>no</td>
</tr>
<tr>
<td>Yahoo! Slurp</td>
<td>yes</td>
<td>remove from index</td>
<td>no</td>
</tr>
</tbody>
</table>

in order to conduct themselves ethically. The additional cases include the following: (1) If a crawler encounters errors in robots.txt files but still parses and obeys the intention of the rules while crawling the site, it is considered more ethical than crawlers that ignore the errors. (2) Webmasters sometimes change the robots.txt rules during a crawl. Accessible directories may be restricted after the change. If a search engine respects the updated rules, it is considered more ethical than those that ignore the updates. (3) Many robots.txt files contain variations of robot names such as Googlebot/1.0. If a crawler notices the variations, it is considered more ethical than those that ignore the variations. These behaviors are not measurable for all crawlers. Thus, we only show the results for known search engine crawlers (see Table 6.9).

6.2.3 Temporal Ethicality

The changes of crawler behavior in the temporal domain are also captured in our research. Figure 6.10 shows both probabilistic and relative ethicality as a function of access date.

Only selected crawlers of interest are shown in the temporal plots. Logarithmic scale is used in the probabilistic ethicality plot because the score varies too much for a linear scale to capture the differences. The temporal ethicality results show that the behavior of crawlers changes significantly over time. The changes are more likely due to the inlink of the honeypot and the update policy of crawlers. More temporal data will be collected in future work for a detailed analysis of this.
Table 6.10. Comparison of crawler probabilistic ethicality and favorability scores.

<table>
<thead>
<tr>
<th>Crawler</th>
<th>Probabilistic Ethicality</th>
<th>Favorability</th>
</tr>
</thead>
<tbody>
<tr>
<td>hyperestraier/1.4.9</td>
<td>0.044444</td>
<td>-0.000006</td>
</tr>
<tr>
<td>msnbot-media/1.0</td>
<td>0.005435</td>
<td>-0.00022</td>
</tr>
<tr>
<td>googlebot-image/1.0</td>
<td>0.002510</td>
<td>-0.002575</td>
</tr>
<tr>
<td>gigabot/3.0</td>
<td>0.002056</td>
<td>-0.000622</td>
</tr>
<tr>
<td>yahoo! Slurp</td>
<td>0.001596</td>
<td>0.031618</td>
</tr>
<tr>
<td>googlebot/2.1</td>
<td>0.001358</td>
<td>0.048683</td>
</tr>
<tr>
<td>charlotte/1.0b</td>
<td>0.000067</td>
<td>-0.00011</td>
</tr>
<tr>
<td>msnbot/1.0</td>
<td>0.000035</td>
<td>0.048683</td>
</tr>
<tr>
<td>ask jeeves/teoma</td>
<td>0.000000</td>
<td>0.005369</td>
</tr>
<tr>
<td>ia_archiver</td>
<td>0.000000</td>
<td>-0.059123</td>
</tr>
</tbody>
</table>

6.2.4 Compare to Favorability

Websites can explicitly specify an access preference for each robot by name. Our previous study formally defined the bias and proposed a favorability measure to evaluate the bias toward crawlers from 2.2 million websites [69]. Since most of the crawlers visiting our honeypot were also being analyzed in the bias study, we list the favorability scores of 10 crawlers with their probabilistic ethicality (see Table 6.10).

The results show that there is no significant correlation between the ethicality and favorability measure of crawlers, which means that more ethical crawlers may not necessarily be more favored.

6.3 Applications

We propose an automated system to monitor the ethicality of web crawlers based on their access behavior with respect to the Robots Exclusion Protocol. Figure 6.11 illustrates the system
architecture.

![Figure 6.11](image)

**Figure 6.11.** A rule based system to monitor the ethicality of web crawlers.

The system relies on the analysis of crawler-generated log records and the ethicality results from section 5. A violation table is first constructed by calculating ethicality vectors based on the log analysis. The ethicality vector for each crawler is then coupled with the robots.txt file of the system website. The webmaster can monitor the ethicality of web crawlers through the system. A threshold ethicality score can be set to generate automatic warning reports for incoming crawler visits. The system can be used to prevent unethical crawlers as well as to regulate general web traffic generated by crawlers.

### 6.4 Effectiveness of Search Engines

The major advantage for websites allowing search engine crawlers to collect their web pages is that the search engines will bring traffic back to the websites. From this perspective, being ethical for a web crawler means bringing the amount of visits corresponding to their crawling visits. Thus, the effectiveness of search engines to a website can be studied by comparing the visits from search engines to the visits generated by correspondingly crawlers.

To measure the effectiveness of search engines, we define return ethicality as the ratio between the visits referred by a search engine and the visits generated by the crawlers of the search engine:

\[
E_{\text{return}}(S) = \frac{\text{Referenced}(S)}{\text{Crawled}(S)}
\]

(6.7)

where \( \text{Referenced}(S) \) is the visits referred from search engine \( S \), and \( \text{Crawled}(S) \) is the crawler visits from search engine \( S \). Based on this measure, the effectiveness of search engines is tested on four different websites in a 40-day period. The effectiveness measure of the above-referred search engines is shown in Table 6.12. The data were collected between 2008/05/13 and 2008/06/21. Website #1 is CiteSeer, a large-scale academic digital library for computer science. Website #2 is a Chinese movie information and retail website. Website #3 is guopi.com, a Chinese makeup retail store. Website #4 is BotSeer, a robots.txt and crawler search engine.

We are more interested in comparing the big search engines. The first three websites are well established websites. Thus we compare the effectiveness measure of Google, Yahoo, MSN
Table 6.11. Effectiveness of search engines.

<table>
<thead>
<tr>
<th>Website #1</th>
<th>Crawled</th>
<th>Referenced</th>
<th>$E_{\text{return}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>16799253</td>
<td>260898</td>
<td>0.01553</td>
</tr>
<tr>
<td>yahoo</td>
<td>17375962</td>
<td>3919</td>
<td>0.00023</td>
</tr>
<tr>
<td>msn</td>
<td>677181</td>
<td>362</td>
<td>0.000535</td>
</tr>
<tr>
<td>baidu</td>
<td>27</td>
<td>37</td>
<td>1.37037</td>
</tr>
<tr>
<td>ask</td>
<td>101912</td>
<td>25</td>
<td>0.00025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Website #2</th>
<th>Crawled</th>
<th>Referenced</th>
<th>$E_{\text{return}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>yodao</td>
<td>632163</td>
<td>81474</td>
<td>0.12888</td>
</tr>
<tr>
<td>baidu</td>
<td>622667</td>
<td>61964</td>
<td>0.09951</td>
</tr>
<tr>
<td>google</td>
<td>872001</td>
<td>46469</td>
<td>0.05329</td>
</tr>
<tr>
<td>sogou</td>
<td>1297047</td>
<td>17946</td>
<td>0.01384</td>
</tr>
<tr>
<td>soso</td>
<td>224322</td>
<td>6847</td>
<td>0.03052</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Website #3</th>
<th>Crawled</th>
<th>Referenced</th>
<th>$E_{\text{return}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>baidu</td>
<td>1830847</td>
<td>844786</td>
<td>0.46142</td>
</tr>
<tr>
<td>google</td>
<td>368417</td>
<td>145115</td>
<td>0.39389</td>
</tr>
<tr>
<td>sogou</td>
<td>462632</td>
<td>62985</td>
<td>0.13615</td>
</tr>
<tr>
<td>yodao</td>
<td>112247</td>
<td>19089</td>
<td>0.17006</td>
</tr>
<tr>
<td>yahoo</td>
<td>315119</td>
<td>11819</td>
<td>0.03751</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Website #4</th>
<th>Crawled</th>
<th>Referenced</th>
<th>$E_{\text{return}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>308529</td>
<td>6478</td>
<td>0.021</td>
</tr>
<tr>
<td>yahoo</td>
<td>36523</td>
<td>160</td>
<td>0.00438</td>
</tr>
<tr>
<td>msn</td>
<td>16589</td>
<td>46</td>
<td>0.00277</td>
</tr>
<tr>
<td>majestic12</td>
<td>3025</td>
<td>6</td>
<td>0.00198</td>
</tr>
<tr>
<td>web-sniffer</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

The results show that the effectiveness of search engines varies significantly for different websites. Ranking based on the referred visits, *Google* plays a dominant role in the two U.S.-based websites and ranks #2 and #3 in the two China-based websites. *Yahoo* and *MSN* have obviously been left behind in the Chinese market. *Baidu* is still largely focused on the market in China.
Figure 6.12. Comparison of the effectiveness of Google, Yahoo, MSN and Baidu.

<table>
<thead>
<tr>
<th></th>
<th>Crawled</th>
<th>Referenced</th>
<th>$E_{return}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>16799253</td>
<td>260898</td>
<td>0.01553</td>
</tr>
<tr>
<td></td>
<td>872001</td>
<td>46469</td>
<td>0.05329</td>
</tr>
<tr>
<td></td>
<td>368417</td>
<td>145115</td>
<td>0.39389</td>
</tr>
<tr>
<td>yahoo</td>
<td>17375962</td>
<td>3919</td>
<td>0.00023</td>
</tr>
<tr>
<td></td>
<td>502584</td>
<td>1249</td>
<td>0.00249</td>
</tr>
<tr>
<td></td>
<td>315119</td>
<td>11819</td>
<td>0.03751</td>
</tr>
<tr>
<td>msn</td>
<td>677181</td>
<td>362</td>
<td>0.00054</td>
</tr>
<tr>
<td></td>
<td>16330</td>
<td>5448</td>
<td>0.33362</td>
</tr>
<tr>
<td></td>
<td>51128</td>
<td>3801</td>
<td>0.07434</td>
</tr>
<tr>
<td>baidu</td>
<td>27</td>
<td>37</td>
<td>1.37037</td>
</tr>
<tr>
<td></td>
<td>622667</td>
<td>61964</td>
<td>0.09951</td>
</tr>
<tr>
<td></td>
<td>1830847</td>
<td>844786</td>
<td>0.46142</td>
</tr>
</tbody>
</table>

Table 6.12. Comparison of the effectiveness of Google, Yahoo, MSN and Baidu.
Chapter 7

Estimating Crawler Population

Since web crawlers are major contributors to web traffic, it is important to have comprehensive data about web crawlers. With the development of computer technologies, building a web crawler becomes a relatively easy task. There are hundreds of open source web crawler projects available online as well as homemade web crawlers developed for specific services and research projects. One fundamental question is: how many web crawlers are there?

Estimating the size of a population is always an interesting task. Capture-recapture (CR) models have a rather long history in the biometry literature where they have been used to estimate the population sizes of wildlife animals [57, 48, 54, 55]. CR models are also used to estimate the size of the web [20, 38] and telephone lines [56]. In this thesis, the maximum likelihood (ML) method presented in [48] is adopted to estimate the population of web crawlers.

7.0.1 Capture-Recapture Models

We use the capture-recapture method to estimate the size of active web crawlers. This method is valuable in the situation where observing all individuals in a population is not possible. In the original method, researchers visit a target area and set traps to capture a group of individuals of a kind of animal. These individuals are marked with unique identifications and released back into the environment. Next, the researchers capture another group of individuals from the same population. The population size can be estimated by applying statistical models to the two data sets obtained from the capture-recapture experiments.

The experiments of estimating web crawler population have similar settings to the wildlife animal study. The World Wide Web can be considered as a wild field for web crawlers. A web site can be seen as a trap to crawlers where researchers can “capture” crawlers and “mark” them by analyzing the access logs. Each web crawler corresponds to an individual in the population that may or may not crawl a web site. In this setting, access logs for a period of time is a sampling occasion in which web crawlers are marked and identified by IP addresses and User-Agent strings. For the access log of a given web site, we can obtain a series of sampling occasions by segmenting
the log with a fixed time interval. We can then apply capture-recapture models on the sampling occasions to estimate the size of web crawlers.

### 7.0.1.1 Lincoln-Peterson Model

The simplest statistical model is the Lincoln-Peterson model based on three assumptions:

- the population to be estimated is closed
- each capture occasion is independent
- individuals have the same probability of being captured.

Let $C_1$ and $C_2$ be the two groups of individuals captured in the first observation and second observation respectively. Denote by $|C|$ the number of elements of the set $C$. The probability of $|C_1|$ individuals being captured is:

$$P(C_1) = \sum_{i \in C_1} \frac{1}{N} = \frac{|C_1|}{N} \tag{7.1}$$

where $N$ is the size of the population. Since $C_1$ and $C_2$ are obtained independently, we have:

$$P(C_1) = \frac{P(C_1 \cap C_2)}{P(C_2)} = P(C_1 \cap C_2 | C_2) \tag{7.2}$$

$P(C_1 \cap C_2 | C_2)$ can be estimated by counting the overlap between $C_1$ and $C_2$:

$$P(C_1 \cap C_2 | C_2) \sim \frac{|C_1 \cap C_2|}{|C_2|} \tag{7.3}$$

Therefore, the population size is given by Equation 7.4.

$$N = \frac{|C_1|}{P(C_1)} = \frac{|C_1|}{P(C_1 \cap C_2 | C_2)} = \frac{|C_1|}{|C_1 \cap C_2|} \tag{7.4}$$

Let’s consider an experiment with 2 capture occasions $t = 2$, $n_1 = 827$, $n_2 = 241$, $m_2 = 24$ and $M_3 = 827 + 241 - 24 = 1044$. The result of the Lincoln-Peterson model estimator is $N = \frac{n_1 n_2}{m_2} = \frac{827 \times 241}{24} = 8,304$.

### 7.0.1.2 Dependency of Capture Sources

#### Two Sources

In the Lincoln-Peterson model, the assumptions of the model are individuals have the same probability of being captured and two capture occasions are independent. If capture occasions $C_1$ and $C_2$ are independent, then

$$P(C_1 | C_2) = P(C_1) \tag{7.5}$$

and

$$P(C_1 | C_2) = \frac{P(C_1 \cap C_2)}{P(C_2)}. \tag{7.6}$$
Thus,

\[ C = \frac{P(C_1 \cap C_2)}{P(C_1) \cdot P(C_2)} = 1. \]  
(7.7)

where \( C \) is defined to be the independence factor.

For the two capture sources:

\[
\begin{array}{c|cc}
& \text{in } C_1 & \text{not in } C_1 \\
in C_2 & a & b \\
not in C_2 & c & x \\
\end{array}
\]

\[
C = \frac{P(C_1 \cap C_2)}{P(C_1) \cdot P(C_2)} \quad (7.8)
\]

\[
= \frac{\frac{a}{N}}{\frac{(a+b)}{N} \cdot \frac{(a+c)}{N}} \quad (7.9)
\]

\[
= \frac{a \cdot N}{(a+b)(a+c)} \quad (7.10)
\]

Since \( N = a + b + c + x \), we have

\[
C = \frac{a + b + c + x}{a + b + c + \frac{bc}{a}}. \quad (7.11)
\]

Therefore, when \( xa = bc \), the two sources are independent and the population \( N \) is correctly estimated. If \( C > 1 \), then \( xa > bc \), thus the population \( N \) is underestimated. If \( C < 1 \), then \( xa < bc \), thus the population \( N \) is overestimated. Since \( x \) is always unknown for only two sources, the dependency of two sources can never be tested.

**Three Sources** Figure 7.1 illustrates the statistics of three capture sources.

Figure 7.1. Set drawing of three capture sources \( C_1, C_2 \) and \( C_3 \).
As an example, let’s test the dependency between two sources $C_1$ and $C_2$.

\[ C = \frac{P(C_1 \cap C_2 | C_3)}{P(C_1 | C_3) \cdot P(C_2 | C_3)} \]  \hspace{1cm} (7.12)

\[ = \frac{\frac{e}{e+f+g+c}}{\frac{e+f+g+c}{e+f+g+c}} \cdot \frac{\frac{e+g}{e+f+g+c}}{\frac{e+g}{e+f+g+c}} \]  \hspace{1cm} (7.13)

\[ = \frac{e + f + g + c}{e + f + g + \frac{fg}{e}} \]  \hspace{1cm} (7.14)

\[ = \frac{e + f + g + \frac{fg}{e}}{e + f + g + \frac{fg}{e}} \]  \hspace{1cm} (7.15)

Obviously, if capture sources $C_1$ and $C_2$ are independent from each other, then $C = 1$. Thus $ec = fg$.

Therefore, when $ec = fg$, the two sources $C_1$ and $C_2$ are independent and the population $N$ is correctly estimated. If $C > 1$, then $ec > fg$, thus the population $N$ is underestimated. If $C < 1$, then $ec < fg$, thus the population $N$ is overestimated.

For three capture sources, $e, c, f, g$ can all be observed from the experiments. Therefore, it is possible to test the dependency relationships between captures with multiple sources.

### 7.0.1.3 Model $M_0$

More generally, we can present the capture-recapture model $M_0$ using maximum likelihood estimator. $M_0$ has the same assumptions as the Lincoln-Peterson model. Let $k$ be the number of capture occasions (observations), $m_j$ be the number of marked individuals in a population at the $j^{th}$ capture occasion ($j = 1, ..., k$), $N$ be the true population size, $\hat{N}$ be the estimator of population size, $n_j$ be the number of individuals captured in the $j^{th}$ capture occasion, $M_{k+1}$ be the number of different individuals caught during all occasions and $p$ be the probability of an individual being captured. The likelihood function for recapture is:

\[ L(N, p) = \frac{N!}{(N - M_{k+1})!} \cdot p^{\sum_j n_j} \cdot (1 - p)^{kN - \sum_j n_j} \]  \hspace{1cm} (7.16)

There are two parameters in the model, $N$ and $p$. $\hat{p}$ is given by the zero point of the partial derivative of the log-likelihood function:

\[ \ln L(N, p) = \ln \left(\frac{N!}{(N - M_{k+1})!}\right) + \sum_j n_j \ln(p) + (kN - \sum_j n_j) \ln(1 - p) \]  \hspace{1cm} (7.17)

\[ \frac{\partial \ln L(p, N)}{\partial p} = 0 \]  \hspace{1cm} (7.18)

\[ \hat{p}(N) = \frac{\sum_j n_j}{kN} \]  \hspace{1cm} (7.19)

The variance of parameter $N$ is given by:
\[ \text{Var}(\hat{N}) = \frac{\hat{N}}{(1 - \hat{p})^{-k} - k(1 - \hat{p})^{-1} + k - 1} \]  

(7.20)

Therefore, the population size \( N \) can be estimated by searching \( N \) values on \( N > M_{t+1} \).

Let’s consider an experiment with 2 capture occasions \( k = 2 \), \( n_1 = 827 \), \( n_2 = 241 \), \( m_2 = 24 \) and \( M_3 = 827 + 241 - 24 = 1044 \), the log-likelihood estimator gives the distribution of population size \( N \) as shown in Figure 7.2. The \( N \) value that maximize the likelihood function is at \( N = 11,844 \).

![Figure 7.2](image)

Figure 7.2. The distribution of population size \( N \) with model \( M_0 \). The maximum likelihood is obtained at \( N = 11,844 \).

The log-likelihood function shows that the variance is much larger than the \( N \) value for model \( M_0 \) which means the estimation accuracy for this model is low.

### 7.0.1.4 Model \( M_h \)

Our goal is to accurately estimate the size of web crawlers. The Lincoln-Peterson model is simple and intuitive. However, the assumptions are too strong. It is very unlikely that all web crawlers have the same probability being captured in a website. More advanced models should be adopted to consider the heterogeneity and other properties of web crawler population.

A finite mixture model \( M_h \) developed in [54] considers the heterogeneity of individuals. It assumes that each individual belongs to one of \( A \) groups with probability \( \pi_a, a = 1, ..., A \) (\( \sum \pi_a = 1 \)) and individuals in group \( a \) have the same probability \( \theta_a \) of being captured in each capture occasion. Thus, for \( k \) capture occasions, the probability of a web crawler being captured \( j \) times is:

\[ \sum_{a=1}^{A} \pi_a \theta_a^j (1 - \theta_a)^{k-j}. \]  

(7.21)
Let $f_j$ be the number of crawlers that are captured exactly $j$ times ($j \in [1, k]$). The likelihood for the model is written as

$$L(N, \pi, \theta|\text{data}) = \frac{N!}{(N - M_{t+1})!} \prod_{j=1}^{k} \left( \sum_{a=1}^{A} \pi_a \theta^j_a (1 - \theta^j_a)^{k-j} \right)^{f_j}$$

(7.22)

There are $2A + 1$ independent parameters in the model. Therefore, the number of capture occasions $k$ should be larger than $2A + 1$ [56]. The capture-recapture toolkit MARK [75] can be used to fit the heterogeneity model and estimate the population of web crawlers with the Markov chain Monte Carlo (MCMC) method. The population size $N$ as a parameter in model $M_h$ can be simulated to find the maximum (see Figure 7.3). The variation of $N$ in model $M_h$ is significantly smaller than in model $M_0$.

**7.0.1.5 Model $M_t$**

In this thesis, multiple data sources are collected to verify the results. Since crawlers are not crawling each website randomly, the probabilities of capturing a crawler in different websites are not constants. Thus, another model considering the behavioral changes $M_t$ should be combined with $M_h$ [48]. In $M_t$, the capture probability in each capture occasion $j$ is $p_j$. The likelihood can be written as

$$L(N, \pi, \theta|\text{data}) = \frac{N!}{(N - M_{t+1})!} \prod_{j=1}^{k} \left( p_j^{n_j} (1 - p_j)^{N-n_j} \right)$$

(7.23)

The variance is given as
The maximum likelihood is obtained at \( \hat{p}_j = \frac{n_j}{N} \). Model \( M_t \) can be used in fitting the data from multiple capture sources where crawlers behave differently in each capture source. The population size \( N \) as a parameter in model \( M_h \) can be simulated to find the maximum (see Figure 7.4). The variation of \( N \) in model \( M_t \) is significantly larger than in model \( M_h \).

7.0.1.6 Model \( M_{th} \)

This model combines \( M_t \) and \( M_h \) and can be used to estimate complicated capture-recapture experiments [54]. The maximum likelihood of model \( M_{th} \) can be written as

\[
L(N, \pi, \theta|\text{data}) = \frac{N!}{(N-M_{t+1})!} \prod_{i=1}^N \sum_{a=1}^A \pi_a \prod_{j=1}^k \left\{ \theta_j a^{x_{ij}} (1 - \theta_j a)^{1-x_{ij}} \right\}
\]  

(7.25)

To estimate the parameter \( N \) for the model \( M_{th} \), a large number of capture occasion from different sources are needed since there are too many parameters in this likelihood function. It is hard to get more sources of crawler access logs. In practice, we use a simplified \( M_{th} \) model which assumes the capture probability of each crawler is related to the visiting frequency of the crawler. Crawlers are then categorized into 3 groups based on their visiting frequency. We apply model \( M_t \) for each group of crawlers and the combined results is used to estimate the overall population \( N \). The population size \( N \) as a parameter in model \( M_{th} \) can be simulated to find the maximum (see Figure 7.5). The variation of \( N \) in model \( M_{th} \) is smallest among the models.
Figure 7.5. The simulation of population size $N$ in model $M_{th}$.

introduced above which indicates the model is more accurate to estimate the crawler population.
Conclusions and Future Work

This thesis has presented a comprehensive study of crawler regulation and behavior. Quantitative metrics and computational models are proposed to measure crawler biases and ethics. The results provide important resources for policy makers, webmasters and researchers to regulate crawlers and to study issues related to the Robots Exclusion Protocol, including privacy, security, load balancing and computer ethics.

A detailed survey of the usage of the robots.txt files has been presented. The statistics show that more than 30% of websites implemented the Robots Exclusion Protocol to regulate web crawlers; this percentage increased throughout the period of our study. We observe that 46.02% of newspaper websites currently have implemented robots.txt files, and the newspaper domain is the one in which the Robots Exclusion Protocol is most frequently adopted. Of U.S. university websites, 45.93% in our sample adopt the Robots Exclusion Protocol, significantly more than European (37.8%) and Asian (15.4%) university sites. Based on the Robots Exclusion Protocol, the quantitative metric “favorability” is proposed to measure biases toward web crawlers in robots.txt files. Our results show that the crawlers of popular search engines and information portals such as Google, Yahoo and MSN are generally favored by most of the websites we sampled. A larger scale analysis of 4.6 million robots.txt files confirms our findings.

Many incorrect uses of the Robots Exclusion Protocol were found in our data. Because the Robots Exclusion Protocol is not an official standard, the actual implementation of parsing the robots.txt file depends on how the author of a crawler interprets it. For example, Googlebot only matches an exact user agent string and ignores the “Crawl-Delay” rule. But MSNbot and Yahoo Slurp claim to obey this rule. The Robots Exclusion Protocol is also an advisory standard, which means crawlers can choose to ignore the rules. Thus, the actual behavior of crawlers may differ significantly from the regulations. Therefore, crawler-generated visits result in significant ethical issues, which until now have not been studied quantitatively.

The conformance of crawlers to the robots.txt rules reveals ethical issues of web crawlers. In this thesis, we formally define the crawler ethicality as the level of conformance of a crawler...
to the Robots Exclusion Protocol. We use a honeypot, a set of sites configured with distinct regulation specifications, in order to capture specific behaviors of web crawlers with respect to the Robots Exclusion Protocol. Based on the page structure of the honeypot and the specifications in corresponding robots.txt files, a set of quantitative metrics is proposed to automatically measure the ethicality and to some degree the intelligence of web crawlers. A log analysis and rule induction module characterizes crawler behavior, and crawler ethicality scores are computed based on the violation table extracted from the honeypot log analysis. Our crawler ethicality results show that commercial crawlers typically have low ethicality scores which represent less unethical activities. Many commercial crawlers, however, were found to violate some rules of the Robots Exclusion Protocol.

The ethicality of web crawlers is also examined from a cost-benefit perspective where the benefit is user visits referred from a search engine and the cost is the visits generated by the search engine crawler. Thus, the effectiveness of search engines is defined as the ratio of the number of visits referred by a search engine to the number of crawler visits generated by the search engine. Four websites have been analyzed using the effectiveness measure. The results show that major search engines differ in their effectiveness for different websites. The results also show significant regional differences for the same search engine in different regional websites.

The bias and ethicality measures are important for policy makers and researchers to study crawler related issues. We have designed and developed BotSeer, the first robots.txt and crawler search engine and analysis tool to deliver the analysis results of web crawlers to web users. The Web-based information system provides resources and services for researching Web crawlers and trends in Robots Exclusion Protocol deployment and adherence. BotSeer currently indexes and analyzes 4.6 million robots.txt files obtained from 17 million websites, as well as three large Web server logs of real-world crawler behavior and related analyses. BotSeer provides fielded robots.txt search, crawler search, bias analysis, and robot-generated log analysis services to aid the Web crawler related studies and developments. With the large amount of data and fielded index, BotSeer can be used to search the favorability and ethicality of web crawlers, compare web crawlers in terms of how well they are received, study web crawler regulations in a specific group of websites, study the relationships between the crawler regulation and the market, and find new communities and networks based on the preferences for certain web robots. BotSeer serves as a resource for studying the regulation and behavior of Web crawlers as well as a tool to inform the creation of effective robots.txt files and crawler implementations.

8.1 Future Work

8.1.1 Bias Analysis

The bias model proposed in this thesis does not consider the content of each website which means that websites are considered with the same weight contributing to the crawler bias measure. Practically, the content and traffic of websites have a significant impact on crawling policies. Thus,
ignoring content results in under-estimating the bias of websites that have a larger impact on the internet. For example, wikipedia.org disfavoring a crawler should have a larger contribution to the total bias score than a small website, which contains only a few pages, disfavoring a crawler. However, the website weighting strategy is a complicated problem since each web page represents different weights as well. Recent research has been conducted to explore in this direction [36]. The study, however, does not fully examine the bias in terms of crawlable content.

In order to provide a more accurate bias measure of web crawlers, properly designing the weighting model of websites is a key to achieve the goal. The website weighting model has to be carefully designed based on specific needs. As discussed in the introduction chapter, the measurement of bias toward web crawlers serves two goals, 1) provide evidence of bias for comparing crawlable content of different information provider, e.g., search engines, and 2) provide references for crawler administrators to understand how their crawlers are received on the web. The two goals may lead to different modeling of website weights. For example, the focus of the modeling should be on the amount of information presented on each website for the first goal. For the second goal, the weighting strategy may be focused on the level of bias that each website presents.

There are many possible ways to model weighted biases of websites. Link analysis methods including PageRank and HITS algorithms are promising candidates because both algorithms provide weighting scores for web pages. However, computing these metrics is typically expensive in terms of computational resources. Although it is possible to query Google for their normalized PageRank scores, the scores only provide a 11-scale reference so that the real differences between web pages are over simplified.

Besides the requirement of expensive computational resources, ethical issues also play an important role in content related bias measures. Without accessing the restricted content, it is hard to estimate the part of websites that are restricted to a specific group of crawlers. Statistical sampling models such as MCMC may be applied to estimate the impact of restricted content. This problem may also be addressed by considering the research of gray literature.

All the above methods emphasize on computational modeling of bias measures. In addition, bringing human experts to manually check a subset of restricted websites (to crawlers) or applying survey method to get direct feedback from webmasters may also help to better understand crawler biases and develop finer granularity crawler bias models. There have been many discussions on webmaster forums talking about how to design and optimize robots.txt files and allow some crawlers and ban others at the same time. These forums include well known webmaster resource websites such as Webmaster-Talk ¹ and Webmaster World ², as well as search engine optimization forums such as SEO Chat ³. An online survey invites the users from these sources is another possible way to study the bias toward web crawlers with explicit feedback. Furthermore, the text of discussion threads can be analyzed automatically to extract the bias toward web crawlers by modeling the attitude and tone of these forum users. Information extraction and language

¹http://www.webmaster-talk.com
²http://www.webmasterworld.com/
³http://forums.seochat.com
processing techniques can be applied in this type of research to help representing crawler biases. Although certain wording style may help to show the users’ attitude, it is still a hard problem to extract opinions from text automatically.

In this thesis, the correlation between the bias measure and the market share of search engines has been studied as an implication of the bias measure. There may also be other ways to study the implications of crawler biases. For example, the "rich get richer" effect can be investigated by considering the content models discussed above. By examining the web pages that are crawlable to a search engine but not to others, models can be built to estimate the extra traffic brought to the search engine because of these web pages. A possible estimation method is to define search engine coverage based on their crawlable content as one essential property of search engines and compare the coverage among different search engines. Furthermore, a coverage to traffic correlation relationship can be constructed by incorporating search engine traffic data to investigate the impact of crawler biases. However, one barrier of applying this method is to obtain the traffic information which is generally not available to the public. Although Alexa web traffic ranking data can be used as a rough estimate for large search engines, it is hard to obtain such data accurately.

Future work can be done on many advanced models of bias with consideration of the methods introduced above. New bias measures should provide richer implications and resources for studying crawler related issues such as search market and crawler regulation technology development in general.

8.1.2 Crawler Ethics

The ethicality models also have simplification issues where the content of websites is ignored. The content related methods discussed above can also be applied to provide more sophisticated ethicality measures.

Besides the content related methods, crawler ethics can also be measured to a larger extent where the impact of crawlers is considered in the measurements. Considering crawler impact can lead to a more accurate and detailed analysis of the ethical issues such as the privacy concerns and the consequences of violating a REP rule. For example, a small unknown crawler breaking a REP rule and accessing a proprietary web page does not have the same consequences as Googlebot doing the same thing since the later may result in the proprietary web page being exposed to millions of users. However, measuring crawler impact is a hard problem since crawler related information is extremely limited especially with those crawlers without public documentations.

One possible solution is to categorize the crawlers into different groups by the availability of related information and measure each group with different metrics. For example, search engine crawlers including Google, Yahoo, MSN, Ask, AOL, AlltheWeb, Lycos, etc. can be grouped together, and the impact of corresponding crawlers can be measured based on their daily usage and market share. Since public search engines provide search services through public interfaces, the impact can be also measured by issuing sample queries and analyzing the returned results. Open
source crawlers can also be grouped together and apply another set of criteria to measure their impacts. Since most open source crawler projects have a public website to provide downloads and documentations, it is possible to measure the impact based on the popularity of these crawlers as well as their ability to process robots.txt files and handle web pages. Many crawlers do not fall into these two groups. One possible measure of impact of these crawlers is cross-referencing among different websites and define the impact based on their access patterns. This measurement requires a set of carefully designed websites with sufficient complexity to test the access patterns of crawlers. Our honeypot can be extended to meet these requirements. In addition, the impact measure for public search engines and open source crawlers can be treated as class labels. Therefore, machine learning technologies can be applied to classify unknown crawlers based on the similarity of their access patterns to the crawlers with known classes.

Future work should incorporate these methods to develop a more accurate and robust model to measure crawler ethics.

8.1.3 BotSeer Service

BotSeer provides a set of services to help the study of crawler related issues. Current services include the robots.txt search, crawler search, bias analysis and robots.txt validation and generation. Many other services can be implemented within the BotSeer framework to provide more comprehensive data and resources for researchers as well as webmasters. Examples of such services include crawler registration service, crawler filtering service, and historical data of robots.txt files and crawler analysis.

As shown in previous chapters, crawlers may not identify themselves properly. This situation makes the automated crawler regulation extremely hard. A crawler registration service may help in the situation by providing a common ground for webmasters and crawler administrators to communicate more smoothly. The service should include at least three components: crawler names and IP addresses mapping, crawler name registration, and misidentification dispute. The log analysis data which is available through current BotSeer can be used to provide crawler names and IP addresses mapping by referencing to the reverse DNS calls. A registration component can be implemented with similar architecture of the DNS service and initialize the database based on the crawler name and IP addresses mapping. The registration component should allow crawler administrators to register the names of crawlers and update IP addresses associated with the registered names. A misidentification dispute component should be implemented to support the verification of search engine crawlers and allow well established search engine crawlers to reserve and dispute the misuse of their names. With the three basic components, the crawler registration service will be able to provide webmasters with authoritative information about crawlers. Programming interfaces of the registration service can also provide websites the ability to implement advanced crawler regulation and filtering modules.

Crawler filtering is a key problem in many log analysis applications such as traffic and usage data mining, load balancing, and caching. BotSeer currently implements a crawler filtering
module for Tomcat based on crawler names and their corresponding IP addresses to limit the daily access from one crawler or one user. Similar modules can be implemented to work with Apache, IIS and other popular web servers to achieve the same goal. As we discussed earlier, the crawler filtering module can also access the registration service to provide advanced crawler traffic control.

Historical data service for robots.txt files, bias analysis and crawler ethics measuring is useful and important to many web related studies. Several requests have been sent to us to provide historical data of robots.txt files and crawler behaviors since BotSeer was first launched. The historical snapshot of robots.txt files and crawler logs can also serve as evidence to improper access disputes. However, storing historical records for a large amount of files and analysis results is not trivial problem. The historical data service should consider many issues such as data storage structure, granularity of each snapshot and historical data layout. The goal of BotSeer is to index most robots.txt files on the web, which means the storage system should be able to handle approximately 200 million files for each snapshot. A snapshot interval of one day will bring the system to more than 70 billion files. Considering the indexing and analysis results, the storage and retrieval of the robots.txt files will become a serious problem. The historical data service should be carefully designed to handle these issues and provide value added information at the same time.


Vita
Yang Sun

EDUCATION
Ph.D., Information Sciences and Technology, Dec. 2008
The Pennsylvania State University, at University Park
Thesis: A Comprehensive Study of the Regulation and Behavior of Web Crawlers

M.S., Electrical Engineering, Jun. 2001
Institute of Semiconductor, Chinese Academy of Science

B.S., Physics, Jul. 1998
Peking University, Beijing, China

DISTINCTION
BotSeer in press: http://botseer.ist.psu.edu/press.html
- Webmasters May Shape Search Results (ASSOCIATED PRESS)
- Google Favored by Web Admins (PC World)
- Google Favored Over Other Search Engines By Webmasters (Science Daily)

EXPERIENCE
- Determining Bias to Search Engines from Robots.txt Nov. 2005 - Oct. 2007
- Automatic Citation Extraction Nov. 2005 - Oct. 2006

PUBLICATIONS