Scrapers selectively respect `robots.txt` directives: evidence from a large-scale empirical study

Taein Kim, Karstan Bock, Claire Luo, Amanda Liswood, Emily Wenger*
Department of Electrical and Computer Engineering
Duke University

ABSTRACT

Online data scraping has taken on new dimensions in recent years, as traditional scrapers have been joined by new AI-specific bots. To counteract unwanted scraping, many sites use tools like the Robots Exclusion Protocol (REP), which places a `robots.txt` file at the site root to dictate scraper behavior. Yet, the efficacy of the REP is not well-understood. Anecdotal evidence suggests some bots comply poorly with it, but no rigorous study exists to support (or refute) this claim. To understand the merits and limits of the REP, we conduct the first large-scale study of web scraper compliance with `robots.txt` directives using anonymized web logs from our institution. We analyze the behavior of 130 self-declared bots (and many anonymous ones) over 40 days, using a series of controlled `robots.txt` experiments. We find that bots are less likely to comply with stricter `robots.txt` directives, and that certain categories of bots, including AI search crawlers, rarely check `robots.txt` at all. These findings suggest that relying on `robots.txt` files to prevent unwanted scraping is risky and highlight the need for alternative approaches.

1 INTRODUCTION

Web scraping, the process of systematically extracting and downloading information from websites, has been a part of the internet ecosystem since its early days [22]. Scraping is now a critical part of many companies' business models, allowing search engines to optimize results and shopping sites to compare deals, among other purposes. In recent years, scraping has gained new importance as vast troves of internet data are tapped for a novel purpose: training and operating large-scale AI models. Today's large-scale AI models require terabytes of training data [28, 31], for which the internet is an obvious and cheap source. Whitepapers documenting many of today's biggest models, from GPT to Llama and beyond, openly acknowledge the use of web scraping to create training data [5, 10, 13, 17, 42, 43].

The AI demand for web scraping extends beyond training data. AI models for text generation have a pesky tendency to *hallucinate*, confidently stating incorrect information [27, 39]. To make AI-generated text more reliable, AI developers proposed Retrieval-Augmented Generation (RAG), in which large vectorized databases of web content are used to ground AI responses [24, 26, 32]. These databases are collated via large-scale web scraping and, unlike training datasets, are continuously updated. Additionally, AI "agents," generative models with additional capabilities, can deploy bots to fetch web content as part of their workflow [6, 7, 20].

Beyond well-documented copyright and privacy concerns, widespread scraping for any purpose—including for AI training data, RAG, and agent use—can destabilize public, data-rich websites. Numerous writings from the "grey" (non-academic) literature describe sites being taken offline in recent months due to thousands of download requests from scrapers associated with AI companies [14, 25, 36]. These scraping-induced problems have prompted an escalating tactical war, as content-rich websites attempt to moderate or prevent unwanted web scraping [9, 34, 35].

Among the most widely used anti-scraping solution is the Robots Exclusion Protocol (REP), implemented via a `robots.txt` file located at the site root. The REP lets site owners set rules that specify which bots can scrape information from the site, which sub-domains those bots can access, and how long bots must wait between successive page accesses [2]. Recent work [35] showed a significant uptick in the use of and restrictions in `robots.txt` files after the rise of generative AI models around 2022, purportedly in reaction to increased scraping for training data. The REP was codified in 2022, but it is not legally binding, so compliance requires scraper goodwill.

Although `robots.txt` is widely used, little hard evidence exists to prove its efficacy in preventing scraping. Some gray literature suggests that many crawlers, including AI-specific bots, do not respect `robots.txt` [14]. Other articles claim that certain crawlers ignore `robots.txt` files altogether [9, 36], while still others observe that bots disallowed by `robots.txt` sometimes pretend to be other, sanctioned bots to circumvent restrictions [25]. Despite this, a recent work [33] conducted a study of 7 crawlers and observed that all AI bots that promised to respect `robots.txt` did so.

Given the dearth of options for deterring unwanted web scraping and widespread belief that `robots.txt` can help, there is urgent need for a large-scale, controlled experiment evaluating this belief. Such a study will allow content hosts

*Corresponding author: `emily.wenger@duke.edu`

to make informed decisions about whether `robots.txt` will provide their desired level of protection or if alternative deterrence mechanisms are needed.

Our contribution. We conduct the first controlled study of scraper compliance with `robots.txt` at scale. To do this, we collect anonymized web traffic data from a set of 36 websites we control over 40 days. During this period, we conduct two empirical measurement studies on sites' `robots.txt` files to determine if or to what extent web bots, AI and otherwise, comply with `robots.txt` directives.

The first study, conducted on a single site in our dataset with significant web traffic, deployed three versions of a `robots.txt` file with increasingly strict bot directives, each for two weeks. By analyzing how bot behavior changes in response to different `robots.txt` directives, we can determine which directives, if any, effectively deter bot behavior and which bots are more or less respectful of `robots.txt` . The second study, conducted via passive observation of bot behavior on sites with fixed `robots.txt` files, broadens the perspective on the analysis, determining the frequency with which `robots.txt` files are accessed. Along the way, we make observations about bot and scraper behavior in general that we believe the community will find interesting. Our key findings are as follows:

- **Bots are less likely to respect `robots.txt` that employ strict directives,** such as denying access to certain pages. Across our three versions of `robots.txt` , we see compliance decrease as directives become stricter.
- **SEO bots are most respectful of `robots.txt` , while search engine crawlers are among the least.** AI-specific bots like AI assistants and AI data scrapers, fall in between.
- **Observed non-compliance with `robots.txt` among otherwise respectful bots can sometimes be attributed to spoofing,** in which malicious bots present a false user agent to avoid detection.

The rest of the paper is organized as follows. §2 situates our work in the broader landscape of data scraping. §3 describes our data collection process and gives a dataset overview. §4 outlines our `robots.txt` experiments and results. §5 discusses possible confounding variables in our experiments. §6 lists limitations and future work.

2 CONTEXT: WEB SCRAPING AND BOT DETERRENCE

Prior work has studied web scraping in general, as well as the prevalence and properties of `robots.txt` files. In this section, we position our work in the broader landscape of research on data scraping and scraper deterrence.

2.1 Web scraping: a history

Early days. Web scrapers and crawlers are nearly as old as the internet. In this work, crawling refers to the process of systematically accessing links on the web, while scraping is the act of downloading information from each link. The first known web crawler was the World Wide Web Wanderer, created in 1993 by an MIT undergraduate student to measure web growth [22]. Shortly after this, the first crawler-based search engine, JumpStation, premiered, paving the way for future crawler-fueled search engines like Google [37]. The creation of tools such as APIs has allowed for simultaneous scraping and crawling, since APIs enable easy data downloads [18]. Web crawling and scraping now power a number of useful applications, such as search engines, price comparison, and real-time system monitoring.

Web scraping in the AI era. Interest in data scraping has only increased as AI models have grown in size and scope. The operating principle for many AI model trainers is that more training data produces better models [28], and the internet is a free and expansive source of data. To collect data at scale, model trainers can either scrape the data themselves or rely on pre-collected datasets like Common Crawl [3]. Additionally, there are some datasets, like LAION 5B, which only provide URLs, requiring prospective users to scrape the data from the links [40]. Since this practice has become so common, large AI companies have developed new user agents to identify bots scraping data on their behalf [4].

In recent years, other types of AI-related scraping have emerged: retrieval-augmented generation (RAG) and AI web accesses. RAG [24, 26, 32] allows large language models (LLMs) to supplement their outputs with current online information, since their training datasets struggle to produce factual information on recent events and can "hallucinate" information in their responses [27]. RAG relies on a database of continuously updated web links that models can quickly search. Recent innovations enable generative AI "agents" to access websites while completing a user-requested task [6, 7, 20]. Such web accesses—which range from information lookup to inputting or downloading data—typically involve deputizing a scraper bot to perform the task.

2.2 Bot deterrence

Although certain types of web scraping can benefit site owners (e.g. search engine optimization), other types can be harmful. For example, unfettered scraping can scoop up data that, while publicly available, may have copyright or privacy concerns [35]. Furthermore, numerous blog posts and other articles from recent years have documented how high-intensity scraping by presumed AI crawlers have caused site instability [9, 25, 36]. In light of these downsides, many techniques have been proposed to prevent unwanted scraping.

robots.txt field	Description
user-agent	Refers to a bot with self-declared user agent string (e.g. Googlebot).
allow	Subset of site pages that bot with specific user-agent **can** access.
disallow	Subset of site pages bot with specific user-agent **cannot** access.
crawl-delay	Minimum required time between successive page accesses by specific bot.
sitemap	Provides an overview of subdomains on the site.

Table 1. Common fields in robots.txt files. robots.txt *files constructed using these fields specify how bots can interact with a site, see Figure 1.*

```
User-agent: Googlbot
Allow: /
Crawl-delay: 15

User-agent: *
Allow: /allowed-data/
Disallow: /restricted-data/
Crawl-delay: 30

Sitemap: https://X.X.X/sitemap/sitemap-0.xml
```

Figure 1. Example robots.txt file. *This site allows bots with* user-agent *Googlebot to access all subdomains with a* crawl-delay *of 15 seconds. All other bots are given a* crawl-delay *of 30 seconds and can only access data under the* /allowed-data/ *subdomain.*

The Robots Exclusion Protocol emerged in 1994 as a possible solution to unwanted bot activity. As crawlers became prominent on the early web, it became clear that a mechanism was needed to allow "friendly" scrapers and disallow poorly-behaved ones [29]. The Robots Exclusion Protocol (REP) [2] was proposed to fill this gap. The REP asks developers to place a file called robots.txt at their site root, specifying restrictions on certain bots and bot behaviors on their site. Possible bot directives include restricting subdomain access, enforcing a crawl delay, and allowing only bots with certain user-agent strings—see Table 1 and Figure 1. The REP was codified as RFC 9309 by the Internet Engineering Task Force (IETF) [19] and is widely used [35, 41].

Despite its prominence, the REP has significant drawbacks. First, robots.txt is not a legally binding document, so requires the good-will of the parties involved for compliance [1]. Second, it requires web hosts to maintain extensive knowledge of user agents and entities that they wish to exclude from scraping [35]. This places a heavy burden on hosts to maintain awareness of the ever-changing web landscape. Third, since compliance is optional, some crawlers simply ignore robots.txt . Finally, limited work [33] has studied bot compliance with robots.txt , and none have done so at scale, making its efficacy unknown.

Other bot-blocking methods. Other methods to block unwanted scraping behavior vary from mildly intrusive to very disruptive. *CAPTCHAs*, which ask site visitors to solve a hard (e.g. computationally difficult) problem before viewing or retrieving page content, have historically been effective in deterring bots [45, 46]. However, they add significant friction to user interactions with webpages [12], while recent advances in AI make CAPTCHA solving easy [11, 23, 47], both of which make CAPTCHAs less useful. Companies like Cloudflare market *proprietary bot detection and deterrence* methods, which can be purchased [15]. As a last resort, companies can outright *block the IP addresses* of troublesome bots. However, since many bots run on VPNs, IP addresses can be easily recycled, circumventing this approach. A recent work suggests using *reverse proxies* to better regulate traffic to a site [33]. Finally, other articles suggest novel methods like employing a *proof of work* [25] or even a *tarpit* [9], which creates unending fake content for scrapers.

2.3 Prior work on scraping and robots.txt

Numerous studies have considered behaviors of web scrapers [30, 38], but limited work has studied the relationship between scrapers and robots.txt . Longpre et al. [35] performed a longitudinal analysis of robots.txt files on websites often scraped for AI data and found that robots.txt restrictions tightened after the rise of generative AI models like ChatGPT. Restrictions in these robots.txt files vary among AI bots, with large, well-known companies like OpenAI having the most restrictions. Longpre et al. argue that tight controls in robots.txt may degrade the quality of open-source AI training datasets. Concurrent work from [16] similarly observed tightening in robots.txt restrictions since the advent of generative AI tools.

Do AI bots respect robots.txt ? Limited work has focused on whether AI scrapers specifically respect robots.txt . One small-scale study explored whether a robots.txt file was respected by AI bots and found that, of the 7 AI bots that

Data subset	Unique IP addresses	Unique user agents	Avg. bytes scraped per session	Unique ASNs	Total bytes scraped	Total page visits	Unique page visits
All data	231,859	19,250	82,306	8,841	62,713,813,720	761,956	31,665
Known bots	11,291	405	52,612	179	16,706,054,178	317,532	6,347

Table 2. Overview of our dataset. *The top row corresponds to the entire dataset, while the bottom row documents activity associated with well-recognized bots.*

visited their site, 6 promised to respect `robots.txt` and did so [33]. One (Bytespider from Bytedance) did not promise to respect `robots.txt` and did not. While interesting, the scale of this study prevents it from providing a holistic picture of how bots in general interact with `robots.txt` .

Evidence from grey literature further suggests that non-compliance with `robots.txt` is widespread [36]. Again, however, no large-scale empirical study has considered this question. We hypothesize this literature gap is primarily due to data access limitations, since to make statistically meaningful statements about `robots.txt` (non)compliance, one must have access to millions of web logs.

3 OUR DATASET FOR ANALYZING ROBOTS.TXT COMPLIANCE AT SCALE

3.1 Dataset Preparation

To study scraper compliance with `robots.txt` files at scale, we collect and analyze a large dataset composed of network access data from website endpoints managed by our institution. The dataset contains approximately 3.9 million web requests made to a set of 36 websites from February 12 - March 29, 2025. The data include the following fields:

- **Useragent**: A string identifying the entity accessing a website that often includes the browser, operating system, hardware, and version information of the origin of the request. Well-known bots often provide a simpler identifying string in this field (e.g. Googlebot).
- **Timestamp**: The ISO-8601 formatted time of the request.
- **IP Hash**: A one-way cryptographic hash of the web visitor's IP address. This anonymizes IPs for IRB compliance.
- **Autonomous System Number (ASN)**: A number assigned by the American Registry of Internet Numbers (ARIN) to the entity controlling web visitor's IP.
- **Sitename**: The base website accessed.
- **URI path**: The subdomain accessed within the website. The sitename and URI path combined form the whole URL.
- **Status code**: Code site returned in response to request.
- **Bytes**: Bytes collected by web visitor during site interaction (e.g. amount of downloaded content).
- **Referer**: Site from which web visitor was redirected.

Each row of the dataset corresponds to one page access by a web visitor at a given time.

Bot name	Total hits	% of all traffic	GB of data scraped
YisouSpider	121495	15.95	8.23
Applebot	118258	15.52	0.21
Baiduspider	15132	1.99	0.05
bingbot	12900	1.69	0.80
meta-externalagent	12837	1.68	0.87
Googlebot	9103	1.19	0.85
HeadlessChrome	8365	1.1	1.22
ChatGPT-User	3029	0.4	0.98
yandex.com/bots	2179	0.29	0.28
SemrushBot	2111	0.28	0.06
GPTBot	1225	0.16	0.25
dotbot	1066	0.14	0.01
Amazonbot	1009	0.13	0.07
AhrefsBot	862	0.11	0.02
SkypeUriPreview	831	0.11	0.09
facebookexternalhit	785	0.1	0.05
BrightEdge Crawler	736	0.1	0.06
Scrapy	726	0.1	0.19
ClaudeBot	684	0.09	0.09
Bytespider	561	0.07	0.08

Table 3. Over 40% of web accesses in our dataset are attributable to just 20 bots. *We document the number of unique web accesses for each bot, the proportion of web traffic they compose, and the amount of data each scraped during the 40 day period.*

Data preprocessing. Data from any logged-in institutional users is filtered out before it reaches our servers to protect user privacy. We inspect data after collection and remove several IP hashes associated with internal vulnerability scanning tools and similar entities that are not relevant to our analysis. We screen out 3 IP hashes associated with 294, 362 web accesses through this approach. Then, we *map ASNs to ARIN info* and *standardize bot names* to enhance the data and simplify analysis.

To map ASNs to ARIN info, we leverage the external library whoisit[1] to poll for whois information for all unique ASNs in our incoming dataset. The returned whois information contains information held by ARIN about the registered user behind the ASN, including the declared entity name and description. We also standardize bot names via fuzzy string

[1]https://pypi.org/project/python-whois/

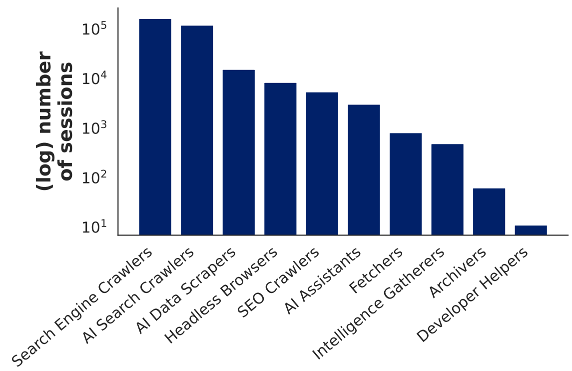

Figure 2. Traditional and AI search, as well as AI data scrapers, are the most active bot types in our dataset. *Headless browsers—a browser running sans GUI, commonly used by scrapers—are fourth.*

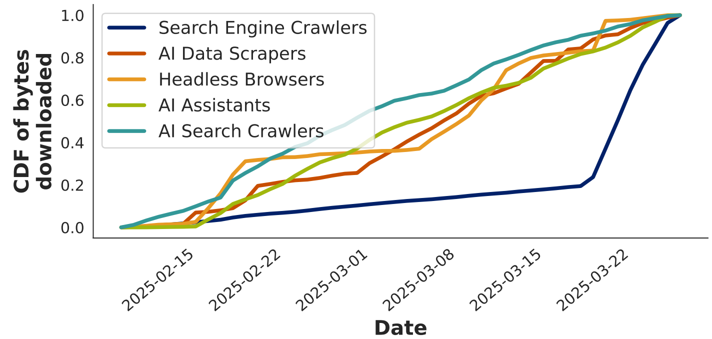

Figure 3. Most bot in the top 5 categories in terms of data scraped collect data steadily, but search engine crawlers buck the trend, driven by YisouSpider's mid-March activity. *AI assistants scrape much data relative to session count (Fig 2).*

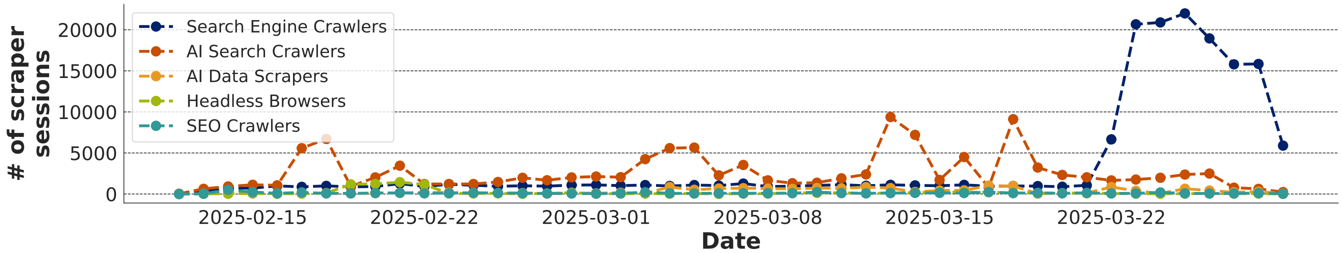

Figure 4. Traditional and AI search crawlers exhibit the most volatility in scraping patterns during the date range encompassed by our dataset. *The volatility in these two categories corresponds directly with high scraping activity of YisouSpider (a search engine crawler) and AppleBot (an AI search crawler). We plot the behaviors of the top* 5 *categories of bots by session count for simplicity.*

matching with a public dataset of common useragent strings and the corresponding bot names[2]. Using these standardized names, we add an additional column to our dataset that maps bots to bot categories proposed by Dark Visitors, an industry website that keeps an up-to-date list of known scrapers[3]—AI Agents, AI Assistants, AI Data Scrapers, Archivers, Developer Helpers, Fetchers, Headless Agents, Intelligence Gatherers, Scrapers, Search Engine Crawlers, SEO Crawlers, Uncategorized, and Undocumented AI Agents. This categorization allows us to analyze collective bot behaviors.

Ethical considerations. Data collection and analysis adhered to a study protocol approved by our local Institutional Review Board (IRB)[4]. We followed all IRB guidelines for anonymizing and storing this dataset, including making significant efforts to exclude non-bot behavior from the dataset and preserve user privacy. We removed sessions originating from logged-in users from our institution or from institution-hosted WiFi networks, and anonymized IP addresses by hashing them. All data was stored and analyzed on secure servers only accessible by the research team.

[2]https://github.com/monperrus/crawler-user-agents
[3]https://darkvisitors.com/
[4]IRB 2025-0094

3.2 Dataset overview

First, we present an overview of our dataset, examining both basic dataset statistics and bot behaviors of interest. Such an overview, independent of our work on `robots.txt` compliance, is valuable. Little academic work has documented the contours of web scraper behavior in the AI age—and none have done so at scale. For our analysis, we aggregate dataset rows, each of which describes a single page access, into time-based "sessions" associated with the same web agent. To do this, we identify accesses by a particular entity/user agent to a set of related pages at contiguous time steps and collapse them into one row, retaining information about individual subdomains visited in a session. We say a session "ends" after 5 minutes of inactivity from an entity. This reduces our dataset from 3,914,096 rows to 761,956 rows.

High-level statistics. Table 2 gives an overview of our dataset, distinguishing between two subsets of data: data associated with any user agent (including generic ones), and data associated with known user agents with defined bot purposes. As the table shows, our dataset contains thousands of unique IP addresses and user agents, but only a handful are associated with known user agents. We define "known bots" as those with self-declared user agents that match either a user agent string format from our Github dataset using regular expression pattern matching, or one recorded by Dark Visitors. The low number of known bots relative to the total unique user agents in our dataset presages the difficulty of determining bot compliance with `robots.txt` , since many operate anonymously or with generic user agents.

Most active bots. Now, we focus specifically on known bots, e.g. bots with self-declared user agents that are well-documented by several web sources. Table 3 gives details about the behavior of the 20 most active bots (as measured by number of web accesses) seen in our dataset. Together, these 20 bots make up over 40% of the observed web traffic. YisouSpider and AppleBot dominate, collectively driving 30% of all traffic and scraping over 8 GB of data during the 40 day period. YisouSpider is associated with the Chinese search engine Yisou and appears to be used to index web pages [44]. AppleBot is a generic bot associated with Apple and is “used by Apple to index search results that allow the Siri AI Assistant to answer user questions” [8].

We also investigate which categories of bots most frequently visit our sites, using the categories defined by Dark Visitors (see §3.1). Figure 2 plots the relative frequency of different types of known bots in our dataset, separated into the categories defined by Dark Visitors. Search-related bots, either for traditional or AI-powered search, are the most frequent visitors, followed by AI data scrapers. Headless browsers, a category assigned to entities observed running a browser without a GUI, come in forth. This category is mostly composed of likely scraper bots that do not identify themselves with a known user agent string. Bots used for other applications, like SEO and archivers, are less common.

Cumulative bot behaviors. Finally, we explore how bots behave over time. Figure 3 plots the CDF of data collected by the top 5 categories of bots in terms of bytes scraped. From this, we see that most bots collect data at a steady rate, with search engine crawlers as a notable exception. These have a significant spike in data collection towards the end of our data collection period. Furthermore, we observe that AI assistants, though ranked sixth in of number of total scraping sessions (Figure 2) are ranked fourth in terms of data scraped (Figure 3), indicating that they collect a relatively large amount of data per session.

Figure 4 shows the number of sessions per day for the five categories of bots with the most total sessions. As with the data CDF plot, most categories of bots exhibit relatively stable behavior over time, with the exception of search engine crawlers and AI search crawlers. Upon further inspection, we find these spikes in activity correspond directly with high scraping activity from YisouSpider (a search engine crawler) and AppleBot (an AI search crawler).

4 MEASURING SCRAPER COMPLIANCE WITH ROBOTS.TXT

Now, we turn our attention to our central question: *understanding the extent to which scraper bots comply with* `robots.txt` *directives*. We first describe our methodology, then analyze the results of our experiments.

4.1 Methodology

Over a period of 8 weeks, we deployed four versions of `robots.txt` files on one site controlled by our institution. This site was chosen because of its observed high bot traffic, making it a rich source of data on scraper behavior. Each `robots.txt` file was deployed for two weeks, then swapped out by support staff for the next version. The four versions of the `robots.txt` files have increasingly strict directives, ranging from no meaningful restrictions to full denial of access for certain bots. Using multiple versions of `robots.txt` allowed us to achieve two distinct goals: measuring whether certain `robots.txt` directives were more or less respected by scrapers, and measuring how quickly scrapers adapted to new `robots.txt` restrictions. The files are described below and shown in Figures 5-8.

- **Base version** (Fig. 5): All bots can access all but three pages (`/404, /dev-404-page, /secure/*`). This was our institution’s standard `robots.txt` setup.
- **Version 1** (Fig. 6): All bots retain the same level of access as in the base version, but a 30 second crawl delay is requested between successive page requests. Crawl delay directive is not part of the official `robots.txt` protcol [2] but is widely used, making it a useful directive to study.
- **Version 2** (Fig. 7): Most bots are allowed to access only the `/page-data/*` endpoint, which we observed experimentally to be a common target for scrapers, but are disallowed from all other sites. Eight SEO bots[5] are exempt from these restrictions per our institution’s request.
- **Version 3** (Fig. 8): Most bots are denied access to all page content. The same eight SEO bots as before are exempt.

We validated that each `robots.txt` file was formatted correctly via the Google `robots.txt` parser[6]. Data collection associated with each of these files was conducted mostly in parallel with the data collection process described in the previous section. The one exception was the baseline `robots.txt` data, which was collected in January 2025 before the full dataset collection started. Table 4 summarizes data collected under each `robots.txt` version. Similar numbers of site visits and unique bots are observed across all four `robots.txt` deployments.

[5]Googlebot, Slurp, bingbot, Yandexbot, DuckDuckBot, BaiduSpider, DuckAssistBot, ia_archiver

[6]https://github.com/google/robotstxt

```
User-agent: *
Allow: /
Disallow: /404
Disallow: /dev-404-page
Disallow: /secure/*
```

Figure 5. Original robots.txt file. *Sitename and host fields are in all robots.txt versions but were removed for anonymous submission.*

```
User-agent: *
Allow: /
Disallow: /404
Disallow: /dev-404-page
Disallow: /secure/*
Crawl-delay: 30
```

Figure 6. Version 1 robots.txt file requires a 30 second crawl delay from all bots.

```
User-agent: Googlebot
Allow: /
Disallow: /404
Disallow: /dev-404-page
Disallow: /secure/*

[above block repeated for user agents: Slurp,
bingbot, Yandexbot, DuckDuckBot, BaiduSpider,
DuckAssistBot, ia_archiver]

User-agent: *
Allow: /page-data/*
Disallow: /
```

Figure 7. Version 2 robots.txt file disallows page-data for most bots.

```
User-agent: Googlebot
Allow: /
Disallow: /404
Disallow: /dev-404-page
Disallow: /secure/*

[above block repeated for user agents Slurp,
bingbot, Yandexbot, DuckDuckBot, BaiduSpider,
DuckAssistBot, ia_archiver]

User-agent: *
Disallow: /
```

Figure 8. Version 3 robots.txt file disallows all pages for most bots.

robots.txt **version**	**unique site visits**	**unique bot visitors**
Base (Fig. 5)	113, 601	78
v1 (Fig. 6)	111, 206	87
v2 (Fig. 7)	119, 399	77
v3 (Fig. 8)	113, 060	75

Table 4. Summary statistics of web traffic captured under all four robots.txt versions. *Site traffic and number of unique bot visitors (bots with known user agents) remains consistent across all robots.txt deployments.*

Data preparation. We performed basic preprocessing on our datasets after collection to aid analysis. Much of our preprocessing pipeline for the robots.txt experiments mirrored that described in §3. However, for our analysis of compliance with robots.txt we also filter out bots that accessed the site less than 5 times under any robots.txt version.

4.2 Metrics for robots.txt compliance

For each version of robots.txt with some restrictions (v1, v2, v3), we develop novel metrics to determine if a bot complies with the directive(s), then use traditional statistical methods to measure whether such compliance is statistically significant relative to the bot's default behaviors. Here, we describe our compliance metrics.

Baseline robots.txt. We use web accesses associated with this version of robots.txt as a control group to determine how/if bot behavior changes in response to stricter directives.

Crawl delay compliance (v1). This robots.txt version implements a 30 second crawl delay, the minimum time bots must wait between successive page accesses. To determine compliance with this directive, we first stratified our dataset into sets of accesses associated with a unique triple τ_i = (ASN, IP hash, user-agent), $i \in [0, K]$, where K is the total number of τ tuples in our dataset. This ensured we were correctly matching access logs with their unique requesting entity. We then sort each set of accesses by time and compute the time elapsed between each successive access.

We denote $\delta_{\tau_i} = \{\delta^j_{\tau_i} | j \in [0, n_i]\}$ as the set of access time deltas associated with tuple τ_i, n_i accesses total. Then, the *compliance ratio* for tuple τ_i, C_{τ_i} is defined as

$$C_{\tau_i} = \frac{|\{\delta^j_{\tau_i} | \delta^j_{\tau_i} \geq 30\}|}{n}.$$

In other words, we compute the ratio of access time deltas greater than or equal to 30 seconds to total time deltas for tuple τ_i. If $C_{\tau_i} = 1$, all access time deltas complied with the 30 second crawl delay directive, and if $C_{\tau_i} = 0$, none did. If there was only one access instance associated with a given τ_i triple, we count this as an instance of compliance, so $C_{\tau_i} = 1$.

Bot category	Crawl delay	Endpoint access	Disallow all	Category average
AI Assistants	0.910 (2077)	0.131 (2154)	**1.000** (1126)	0.665
AI Data Scrapers	0.560 (1862)	0.352 (1824)	**0.771** (1822)	0.560
AI Search Crawlers	**0.895** (1157)	0.673 (975)	0.348 (1056)	0.644
Fetchers	**0.927** (252)	0.277 (252)	0.377 (235)	0.525
Headless Browsers	0.036 (2777)	**0.278** (4207)	0.011 (1282)	0.153
Intelligence Gatherers	**0.809** (1120)	0.372 (549)	0.094 (1008)	0.431
Other	**0.486** (5687)	0.119 (4506)	0.016 (4770)	0.220
SEO Crawlers	0.645 (1580)	**0.832** (1217)	0.639 (1268)	**0.704**
Search Engine Crawlers	**0.777** (6379)	0.365 (5934)	0.186 (6409)	0.443
Directive average	**0.618**	0.314	0.261	0.397

Table 5. Bots are most likely to comply with the crawl delay directives, and SEO Crawlers are the most compliant bots overall. *We measure this by computing weighted averages of compliance ratios, weighted by number of bot accesses, for all bots in a particular category. We* **bold** *the highest compliance value in each row to show directive-level trends in compliance and also* **bold** *the highest row and column averages (last row/column).*

After computing C_{τ_i} for the crawl-delay dataset, we run the same compliance ratio calculation on all access time deltas for τ triples in the baseline `robots.txt` dataset, $C_{\tau_j}^{default}$, to determine the base rate of compliance with the 30 second crawl delay directive. We then use a paired z-test for difference in proportions to determine if there is a statistically significant shift in the rate of compliance with the crawl delay directive for bots before/after the crawl-delay in `robots.txt` is enforced. Since a single user agent (a term we use synonymously with bot) can be associated with multiple tau_i tuples—for example, multiple IP addresses declare the same user-agent—we first group tuples based on user agent name, then analyze behavior across all τ_i tuples associated with this user agent.

Endpoint access compliance (v2). This `robots.txt` version restricts bots (except for a group of SEO bots, see §4.1) to only accessing subpages under a single endpoint, `/page-data`, on the site. Computing the ratio of compliance with this directive is more straightforward. For each unique user-agent, we count its accesses to either the `robots.txt` file (which is always allowed) or `/page-data` subpages and compute the ratio of this to the user agent's total page accesses. This compliance ratio C_{τ_i} is 1 if the user-agent only accesses the allowed pages, `robots.txt` and `page-data`, and is near 0 if it consistently accesses many other pages. We then compute an identical compliance metric for user agents under our default `robots.txt` and use this as the baseline for a paired z-test for differences in proportions between the two datasets. Ideally, we should see a significant uptick in compliance when the endpoint directive is in place.

Disallow compliance (v3). The final `robots.txt` version prevents most bots (again, see exceptions in §4.1) from accessing *any* endpoint on the page, although `robots.txt` is always allowed. "Compliance" in this setting is therefore the ratio of `robots.txt` accesses to total page accesses. If the bot behaves, all page accesses should be to `robots.txt` , resulting in a compliance ratio of 1. Again, we compute this metric for both the dataset of disallow accesses and the baseline dataset and use a paired z-test for differences in proportion to measure change in compliance.

4.3 Results and Discussion

In our analysis of `robots.txt` compliance, we consider three main research questions:

- **RQ1:** With which directive are bots most likely to comply?
- **RQ2:** Which category of bots have the highest rate of overall compliance?
- **RQ3:** Do any individual bots exhibit interesting trends?

To answer RQ1 and RQ2, we group bots by their Dark Visitors category and compute the *weighted* average of bot compliance with the 3 directives for category. We weight the average by number of accesses from a particular bot, so the category average is more heavily weighted towards more common bots. We made this choice due to observed uneven distributions of bot accesses in our dataset. For example, some bots never comply with the directive but appear less than 10 times, while others comply at a much higher rate and appear hundreds of times. A weighted average captures this nuanced behavior. Table 5 displays these results.

We begin with RQ1. In Table 5, we **bold** the directive with the highest compliance rate for each bot. From this, we see that 5 categories of bots comply the most with crawl delay, while 2 categories of bots each have highest compliance with endpoint access and disallow all. Furthermore, we compute the average compliance rate for each directive, the last row in the table, and find the highest average compliance with crawl-delay—nearly 2x that of the next-highest, endpoint access. Based on this analysis, we make the following conclusion:

Answer to RQ1: Bots comply most with the crawl delay directive and least with the disallow all directive. This indicates that **bots are less likely to comply with stricter `robots.txt` directives**.

To answer RQ2, we compute the average compliance rate across all directives for each category and list this in the rightmost column of Table 5. From this, we see that:

Answer to RQ2: SEO Crawlers exhibit the highest rate of compliance with `robots.txt` directives, followed closely by AI Assistants and AI Search Crawlers.

SEO Crawlers rely heavily on `robots.txt` files to guide their actions, so their high compliance rate makes sense. The high compliance rates of AI Assistants and AI Search Crawlers are more surprisingly, especially given articles in the grey literature about AI bot *non-compliance* with `robots.txt` [25, 36]. Further exploration of this disconnect is interesting future work. The least respectful bots, in contrast, are Headless Browsers and "Other." Since both these categories serve somewhat as a catchall, bots in this category are more obscure, making noncompliance unsurprising.

Finally, we consider individual bot behavior in answering RQ3. Figure 9 shows the change in compliance ratio from baseline for each directive across the 26 bots with ≥ 5 web accesses under each directive, denoting statistically significant shifts with a red dotted line. Table 6 breaks out these bots by their category and includes information about their sponsoring organization and public promises made to respect `robots.txt`. From Figure 9 we see a decreasing trend in baseline compliance with `robots.txt` directives as we move from crawl delay to disallow all. This indicates that a number of bots complied by default with our 30 second crawl delay. Additionally, several bots make notable efforts to comply with each directive—Amazonbot, ClaudeBot, and GPTBot in particular. Finally, several bots (such as PerplexityBot) explicitly stated they will not respect `robots.txt` (see Table 6) but have somewhat high compliance. This indicates either that they comply by default (which happened frequently for crawl delay) or are more respectful than they claim. There are also bots like BrightEdge that claim to respect `robots.txt` but have low compliance.

Answer to RQ3: There is significant variation in individual bot responses to `robots.txt` directives.

5 EXPLORING POSSIBLE CONFOUNDING VARIABLES IN ROBOTS.TXT ANALYSIS

While the prior section shows that some bots respect `robots.txt` , there are still several issues with relying on `robots.txt` for restricting bot behavior. First, bots may simply not check `robots.txt` frequently enough for changes in `robots.txt` to be effective preventative measures. Second, bots may "spoof" their user agent, pretending to be a bot with high permissions in `robots.txt` (e.g. Googlebot in our situation) to access otherwise-restricted content. In this section, we investigate how likely these two scenarios are in practice, to help practitioners understand practical limits of relying on `robots.txt` to restrict bot behavior.

5.1 Frequency of `robots.txt` updates

Motivation. There were several bots that simply never checked `robots.txt` during our experiments: 9/34 for the crawl delay experiment, and 15/47 for both the endpoint access and the disallow-all experiments. Some of these bots still complied at least somewhat with the directives, but others did not comply at all. Table 7 lists all bots that did not check at least one of our experimental `robots.txt` versions and their rates of compliance for each version.

The case where bots never check `robots.txt` but still comply (e.g. Slackbot, Googlebot-Image, Brightedge Crawler) raises some interesting questions. Do the bots comply without checking `robots.txt` because (1) compliance happens naturally (e.g. directive is permissive enough) or (2) they retrieve `robots.txt` from some other cached source? Without knowledge of crawler internal settings, both points are impossible to verify experimentally. We speculate that when bots comply with the crawl-delay directive without checking, compliance may have occurred naturally. It is unclear how compliance with stricter directives regarding page access could occur by chance, but investigation of the possibility of `robots.txt` caching is out of scope for this paper.

The other interesting pattern in Table 7 is that some bots check `robots.txt` under some conditions but not others (e.g. DuckDuckBot, Bytespider). This raises the question of whether some bots simply check `robots.txt` infrequently. The standard set forth by Google (which other bots often follow) is that bots should check `robots.txt` every 24 hours [21] but some bots may not follow this. Knowing which bots check `robots.txt` less frequently on average could help web hosts determine how long they should expect to wait before bots adjust behavior based on their deployed `robots.txt` and/or know if changing the `robots.txt` file is the right approach for deterring a particular bot.

Methodology. To understand the frequency with which bots check `robots.txt` , we first find additional sites in our

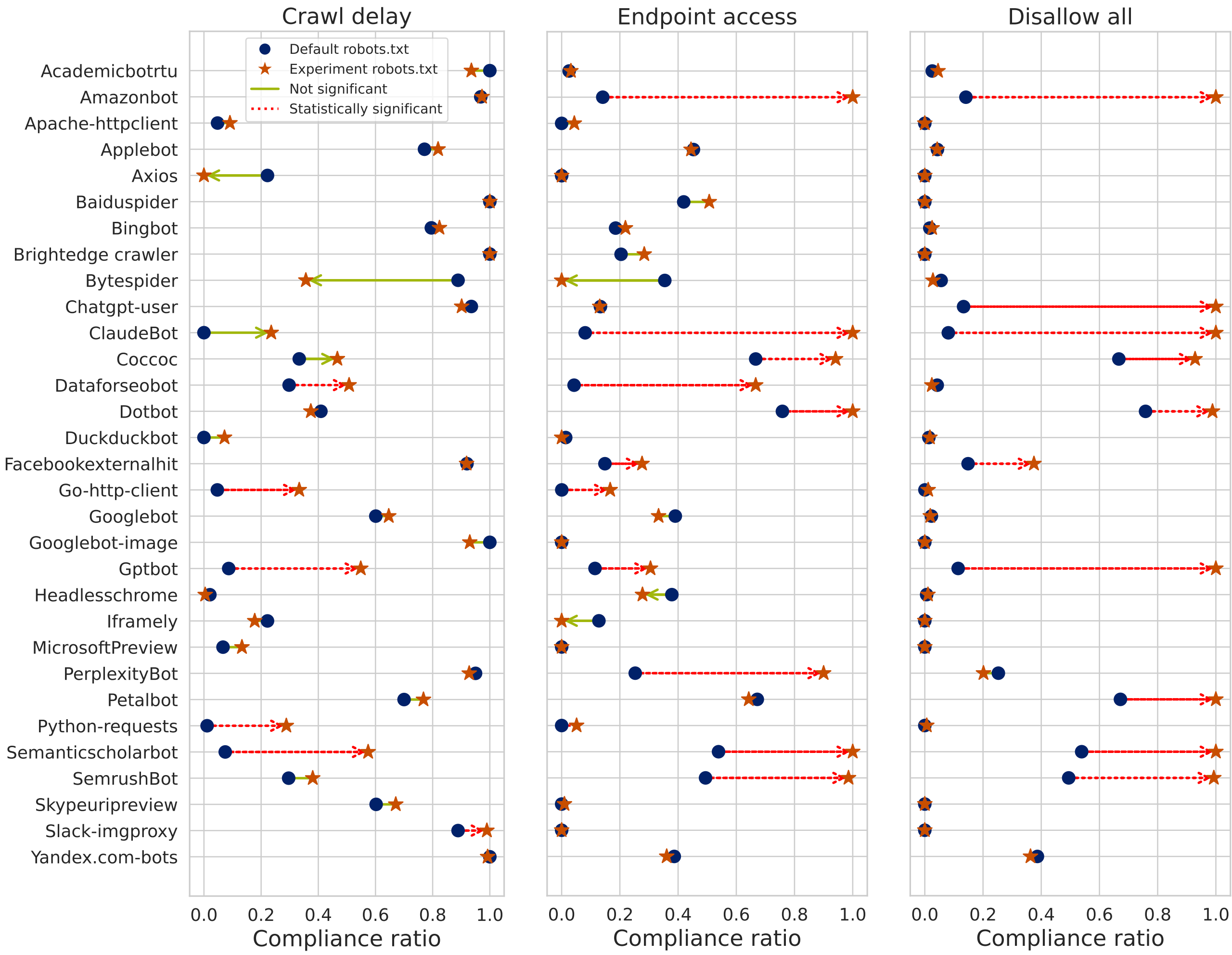

Figure 9. Many bots comply by default with the 30 second crawl delay (leftmost graph), but the rate of compliance decreases overall as `robots.txt` restrictions tighten. *We exclude results for exempted SEO bots and highlight statistically significant ($p \leq 0.05$) shifts with red dotted lines.*

dataset for which `robots.txt` were deployed with meaningful restrictions and were publicly available. Three out of the 36 total sites had `robots.txt` that met this criteria. All three files were identical and implemented simple restrictions on `/404` and `/secure` endpoints on their respective sites. Through dataset inspection, we found that all bots respected these restrictions.

To then analyze the frequency with which bots checked these files, we segmented the access logs for each bot into variable length time windows (12hrs, 24hrs, 48hrs, 72hrs, 168hrs) starting from when the bot first accessed any of these `robots.txt` files. For each time window length, we then counted the number of windows in which the bot accessed `robots.txt` throughout our 40 day dataset. We report a bot as "complying" with a particular time window access length if every time window of that length in the dataset contains a `robots.txt` access. Using the variable length time windows allows us to see the bounds on `robots.txt` frequency—do bots access it every 12 hours? Every 24? Every 168?

We then grouped these results by bot category (e.g. Dark Visitors categories used previously), to give a better sense of aggregate bot behavior, and report the proportion of bots in each category that check `robots.txt` within each of the 5 given time windows in Figure 10.

Results. Some categories of bots (scrapers, archivers, and intelligence gatherers) consistently re-check `robots.txt` within 12 hours if they check it at all. Other categories exhibit more variable behavior, but increasing the window of time between `robots.txt` checks typically leads to a higher re-check rate. AI-related bots (AI Assistants and AI search crawlers) have the lowest re-check rates with less than 40% of bots checking `robots.txt` within a 168 hour window. Thus, frequent

Bot Name	Sponsoring Entity	Bot category	Promise to respect `robots.txt` ?	Crawl delay `robots.txt` compliance	Endpoint `robots.txt` compliance	Disallow `robots.txt` compliance
Academicbotrtu	Riga Technical	Other	Unknown	0.939	0.032	0.045
AhrefsBot	Ahrefs	SEO	Yes	0.697	1.000	1.000
Amazonbot	Amazon	AI Search	Yes	0.973	1.000	1.000
Apache-httpclient	Apache	Other	Unknown	0.130	0.043	0.000
Applebot	Apple	AI Search	Yes	0.841	0.444	0.043
Axios	Open Source	Other	**No**	0.060	0.000	0.000
BrightEdge	BrightEdge	SEO	Yes	1.000	0.284	0.000
Bytespider	ByteDance	AI Data Scraper	**No**	0.398	0.000	0.028
ChatGPT-User	OpenAI	AI Assistant	Yes	0.910	0.131	1.000
ClaudeBot	Anthropic	AI Data Scraper	Yes	0.480	1.000	1.000
Coccoc	Coc Coc	Search Engine	Yes	0.704	0.941	0.929
DataForSEOBot	DataForSEO	SEO	Yes	0.573	0.667	0.024
Dotbot	Moz	SEO	Yes	0.615	1.000	0.988
Facebookexternalhit	Meta	Fetcher	**No**	0.923	0.276	0.375
Go-http-client	Open Source	Other	Unknown	0.474	0.167	0.012
GPTBot	OpenAI	AI Data Scraper	Yes	0.634	0.305	1.000
Headlesschrome	Open Source	Headless Browser	Unknown	0.036	0.278	0.011
Iframely	Itteco	Other	Yes	0.254	0.000	0.000
MicrosoftPreview	Microsoft	Other	Yes	0.294	0.000	0.000
PerplexityBot	Perplexity	AI Search	**No**	0.933	0.900	0.202
PetalBot	Huawei	Search Engine	Yes	0.812	0.643	1.000
Python-requests	Open Source	Other	Unknown	0.462	0.051	0.007
SemanticScholarBot	Allen AI	Search Engine	Yes	0.663	1.000	1.000
SemrushBot	Semrush	SEO	Yes	0.563	0.986	0.993
SeznamBot	Seznam.cz	Search Engine	Yes	0.565	0.833	1.000
SkypeUriPreview	Microsoft	Other	Yes	0.726	0.010	0.000
Slack-imgproxy	Salesforce	Other	**No**	0.917	0.000	0.000
Yandex.com/bots	Yandex	Search Engine	Yes	0.992	0.361	0.363

Table 6. Individual bots respond differently to our `robots.txt` directives. *We plot the top 26 bots with ≥ 5 accesses under each of our `robots.txt` directives. For user agents associated with scraping libraries, we list "open source" as the sponsoring entity.*

updates to `robots.txt` to prevent unwanted scraping may not effectively prevent scraping, particularly for AI bots.

5.2 User agent spoofing

Motivation. While running our analysis in §4, we observed that a few bots had high-but-not perfect compliance rates. For example, Googlebot had a 0.65 compliance rate with the 30 second crawl delay in `robots.txt` , despite checking `robots.txt` several times during this experiment. Similarly, Amazonbot had a 0.96 compliance rate, even though it too regularly checked `robots.txt` . Several other high profile bots like GPT-Bot and bingbot exhibited similar trends.

This led us consider whether bot "spoofing" played a role in lowering measure compliance rate for certain well-known bots. The useragent field in a web access request is manually set by the requesting agent, so malicious bots can easily manipulate this field to misrepresent themselves, potentially circumventing certain `robots.txt` restrictions. Some bots, like Googlebot, have elevated privileges in many `robots.txt` files (including our experiments). By spoofing Googlebot's user agent, a bot could fly under the radar for longer before being banned. They could eventually be detected via IP address checks and behavioral patterns, but spoofing could help them avoid early flagging.

Bot	Crawl delay robots.txt		Endpoint access robots.txt		Disallow all robots.txt	
	Checked robots.txt	Compliance	Checked robots.txt	Compliance	Checked robots.txt	Compliance
Apache-HttpClient	No	0.10	Yes	0.05	No	0.0
Axios	No	0.0	No	0.0	No	0.0
BaiduSpider	No	1.0	No	0.51*	No	0.0*
BrightEdge Crawler	No	1.0	No	0.28	No	0.0
Bytespider	Yes	0.35	No	0.0	Yes	0.02
DuckDuckBot	Yes	0.07	No	0.0*	Yes	0.02*
Googlebot-Image	No	0.98	No	0.0*	No	0.0*
Iframely	No	0.17	No	0.0	No	0.0
Microsoft-Preview	No	0.13	No	0.0	No	0.0
SkypeURIPreview	No	0.67	No	0.01	No	0.0
Slack-ImgProxy	No	0.98	No	0.0	No	0.0

Table 7. Bots that skipped robots.txt check during one or more robots.txt experiments but still complied. * = *bot excluded from endpoint and disallow-all* robots.txt *at institution's request.*

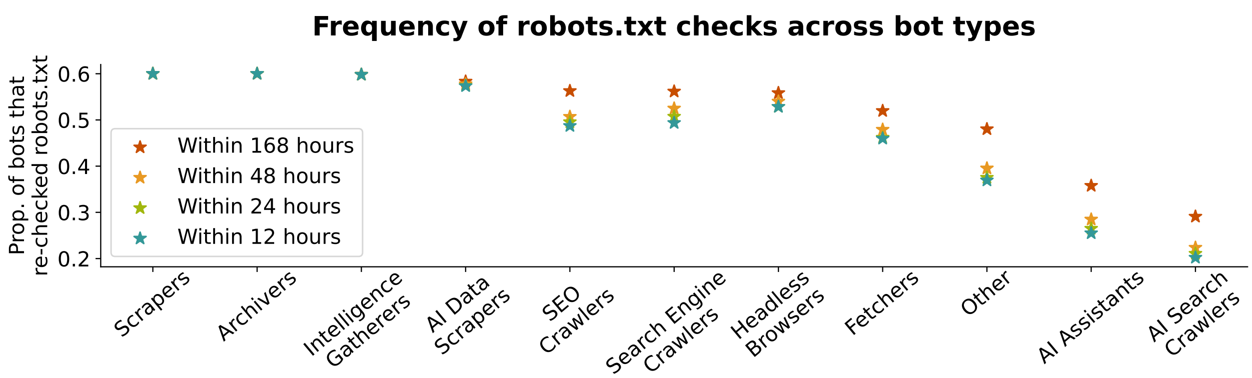

Figure 10. Some bots do not check robots.txt very frequently. *AI assistants and AI search crawlers check* robots.txt *the least.*

Since a spoofing agent may not necessarily abide by the same rules as the benevolent bot, it may not, for example, respect robots.txt . This could plausibly explain the partial non-compliance we saw with some of our robots.txt directives. The grey literature gives some evidence of such spoofing in practice, particularly for large-scale AI bots [25].

Methods. We analyzed web traffic data for known bots and observed empirically that most bots are overwhelmingly associated with a particular ASN. For example, 99.62% of Googlebot's traffic comes from the GOOGLE-CLOUD-PLATFORM ASN, while 99.76% of GPT-Bot's traffic is from MICROSOFT-CORP. Based on this, we develop an empirical heuristic that if a bot's traffic is associated 90% of the time with one ASN, other ASNs associated with this user agent are likely spoofed. Using this heuristic, we analyze web accesses from our dataset and flag any bots associated with more than 1 ASN for which one ASN has over 90% of the traffic. We focus specifically on well-known bots, for which spoofing is more interesting.

Results. Overall, we identify 18 bots for which spoofing may have occurred. In Table 8, we show the flagged bots, the dominant ASN (making up $\geq$ 90% of bot traffic), and the suspicious ASNs ($\leq$ 1% of traffic). Although many flagged bots are only associated with two ASNs, some popular bots such as Googlebot are associated with up to 24. Our analysis cannot prove definitively that these infrequent ASNs belong to spoofers (maybe Google has a contract with Telefonica_de_Espana?), but it strongly suggests this is the case.

There are, on average, less than 5 web accesses associated with these infrequent ASNs for most of the flagged bots. Notable exceptions are Baiduspider, with 381 potentially spoofed accesses out of 15,132; Googlebot, with 33 out of 9103; and SkypeURIPreview, with 26 out of 831. This means that spoofed bots may cause some, but not all, of the noncompliance with robots.txt observed in §4. We believe this relatively low spoofing rate could be due to tools deployed by our institution to filter out potential spoofers using whitelists of known IP addresses for common bots. Since these tools prevent certain suspicious bots from accessing institutional content a priori, we will not observe their traffic in our dataset. Nevertheless, the presence of potential spoofed bots that evade this analytical dragnet is interesting.

Bot	Main ASN (> 90% of accesses)	Possible spoofing ASNs (< 5% of accesses)
AdsBot-Google	GOOGLE	DMZHOST
AhrefsBot	OVH	AHREFS-AS-AP
Amazonbot	AMAZON-AES	CONTABO, DIGITALOCEAN-ASN
Baiduspider	CHINA169-Backbone	CHINAMOBILE-CN, CHINANET-BACKBONE, CHINANET-IDC-BJ-AP, CHINATELECOM-JIANGSU-NANJING-IDC, CHINATELECOM-ZHEJIANG-WENZHOU-IDC, HINET
bingbot	MICROSOFT-CORP-MSN-AS-BLOCK	Clouvider, HOL-GR,MICROSOFT-CORP-AS, ORG-TNL2-AFRINIC, ORG-VNL1-AFRINIC
ClaudeBot	AMAZON-02	GOOGLE-CLOUD-PLATFORM
DuckDuckBot	MICROSOFT-CORP-MSN-AS-BLOCK	DIGITALOCEAN-ASN31, INTERQ31
facebookexternalhit	FACEBOOK	AMAZON-02, AMAZON-AES, KAKAO-AS-KR-KR51
GPTBot	MICROSOFT-CORP-MSN-AS-BLOCK	BORUSANTELEKOM-AS
Google Web Preview	GOOGLE	AMAZON-02
Googlebot-Image	GOOGLE	AMAZON-02
Googlebot/	GOOGLE	52468, ASN-SATELLITE, ASN270353, CDNEXT, CHINANET-BACKBONE, Clouvider, DATACLUB, HOL-GR, HWCLOUDS-AS-AP, IT7NET, LIMESTONENETWORKS, M247, ORG-RTL1-AFRINIC, ORG-TNL2-AFRINIC, P4NET, PROSPERO-AS, RELIABLESITE, RELIANCEJIO-IN, ROSTELECOM-AS, ROUTERHOSTING, TENCENT-NET-AP-CN, Telefonica_de_Espana, VCG-AS
meta-externalagent	FACEBOOK	DIGITALOCEAN-ASN
SkypeUriPreview	MICROSOFT-CORP-MSN-AS-BLOCK	AMAZON-AES, M247
Snap URL Preview Service	AMAZON-AES	AMAZON-02
Twitterbot	TWITTER	PROSPERO-AS, Telegram
yandex.com/bots	YANDEX	AMAZON-02, AMAZON-AES, PROSPERO-AS

Table 8. Popular bots with one dominant ASN and several infreqently-appearing ASNs—a possible sign of spoofing.

6 DISCUSSION

From this large-scale study, we note several limitations of using `robots.txt` as a deterrent for web scrapers, particularly AI scrapers. First, we observe that bots are less likely to comply with more restrictive `robots.txt` directives, such as denying access to certain subpages, but are friendlier about respecting crawl delay. This implies that leveraging `robots.txt` alone to prevent unwanted scraping could be ineffective. Second, we observed that many bots with known user agents in our data **never checked the `robots.txt` file** or **check it infrequently**. This means that any changes made to `robots.txt` to prevent unwanted scraping (e.g. [35]) would not be noticed by the scraper for significant time (if they are respected at all). Finally, we observe numerous instances of potential spoofing of well-known bots, indicating that malicious user-agents may attempt to evade `robots.txt` restrictions by masquerading as bots with higher privileges. Together, these suggest that `robots.txt` does not provide fine-grained access control over web content, highlighting the need for more strongly-enforceable methods to prevent unwanted scraping.

Future work. Numerous threads of future work can build on this study. First, continued work is needed to find low-cost solutions to unwanted data scraping. This could take the form of a legally enforceable standard, a novel technical tool, or another new approach. Second, understanding needs of mid-size web hosts who might be exceptionally targeted by AI scraping (such as libraries and archives) through a user study could surface new, more targeted ways to provide more controlled access to this valuable data. Finally, additional work could dive into the spoofing analysis of §5, determining more robust anti-spoofing methods that catch bots who slip through other enforcement mechanisms.

REFERENCES

[1] 2007. Can a /robots.txt be used in a court of law? https://www.robotstxt.org/faq/legal.html.

[2] 2022. RFC 9309: Robots Exclusion Protocol. https://www.rfc-editor.org/rfc/rfc9309.html.

[3] 2025. Common Crawl. https://commoncrawl.org/.

[4] 2025. Dark Visitors. https://darkvisitors.com/.

[5] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219* (2024).

[6] Open AI. 2025. Introducing Operator. https://openai.com/index/introducing-operator/.

[7] Anthropic. 2025. Clause can now search the web. https://www.anthropic.com/news/web-search.

[8] Apple. 2025. About Applebot. https://support.apple.com/en-us/119829.

[9] Ashley Belanger. 2025. AI haters build tarpits to trap and trick AI scrapers that ignore robots.txt. *Ars Technica* (2025). https://arstechnica.com/tech-policy/2025/01/ai-haters-build-tarpits-to-trap-and-trick-ai-scrapers-that-ignore-robots-txt/.

[10] Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[11] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C Mitchell. 2014. The end is nigh: Generic solving of text-based {CAPTCHAs}. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*.

[12] Elie Bursztein, Steven Bethard, Celine Fabry, John C Mitchell, and Dan Jurafsky. 2010. How good are humans at solving CAPTCHAs? A large scale evaluation. In *2010 IEEE symposium on security and privacy*. IEEE, 399–413.

[13] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.

[14] Thomas Claburn. 2024. Automation needed to fight army of AI content harvesters stalking the web. https://www.theregister.com/2024/07/30/taming_ai_content_crawlers/.

[15] Cloudflare. 2024. https://www.cloudflare.com/application-services/products/bot-management/.

[16] Michael Dinzinger and Michael Granitzer. 2024. A longitudinal study of content control mechanisms. In *Companion Proceedings of the ACM Web Conference 2024*. 1382–1387.

[17] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[18] Roy Thomas Fielding. 2000. Architectural Styles and the Design of Network-based Software Architectures. https://ics.uci.edu/~fielding/pubs/dissertation/top.htm?deviceId=c9100f78-4e17-449f-b247-5f55bf2b13bc.

[19] Internet Engineering Task Force. 2022. RFC 9309. https://datatracker.ietf.org/doc/html/rfc9309.

[20] Google. 2025. Gemini Deep Research. https://gemini.google/overview/deep-research/?hl=en.

[21] Google. 2025. How Google interprets the robots.txt specification. https://developers.google.com/search/docs/crawling-indexing/robots/robots_txt.

[22] Matthew Gray. 1995. Measuring the Growth of the Web. https://www.mit.edu/people/mkgray/growth/.

[23] Meriem Guerar, Luca Verderame, Mauro Migliardi, Francesco Palmieri, and Alessio Merlo. 2021. Gotta captcha'em all: A survey of 20 years of the human-or-computer dilemma. *ACM Computing Surveys (CSUR)* 54, 9 (2021), 1–33.

[24] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*. PMLR, 3929–3938.

[25] Xe Iaso. 2025. Amazon's AI Crawler is Making my Git Server Unstable. https://xeiaso.net/notes/2025/amazon-crawler/.

[26] Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282* (2020).

[27] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys* 55, 12 (2023), 1–38.

[28] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[29] Martijn Koster. 1994. Important: Spiders, Robots and Web Wanderers. https://web.archive.org/web/20131029200350/http://inkdroid.org/tmp/www-talk/4113.html.

[30] Junsup Lee, Sungdeok Cha, Dongkun Lee, and Hyungkyu Lee. 2009. Classification of web robots: an empirical study based on over one billion requests. *computers & security* 28, 8 (2009), 795–802.

[31] Christopher Lehane. 2025. [OpenAI Response] OSTP/NSF RFI: Notice Request for Information on the Development of an Artificial Intelligence (AI) Action Plan. https://cdn.openai.com/global-affairs/ostp-rfi/ec680b75-d539-4653-b297-8bcf6e5f7686/openai-response-ostp-nsf-rfi-notice-request-for-information-on-the-development-of-an-artificial-intelligence-ai-action-plan.pdf.

[32] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[33] Enze Liu, Elisa Luo, Shawn Shan, Geoffrey M Voelker, Ben Y Zhao, and Stefan Savage. 2024. Somesite I Used To Crawl: Awareness, Agency and Efficacy in Protecting Content Creators From AI Crawlers. *arXiv preprint arXiv:2411.15091* (2024).

[34] Shayne Longpre. 2025. AI crawler wars threaten to make the web more closed for everyone. *MIT Tech Review* (2025). https://www.technologyreview.com/2025/02/11/1111518/ai-crawler-wars-closed-web/.

[35] Shayne Longpre, Robert Mahari, Ariel Lee, Campbell Lund, Hamidah Oderinwale, William Brannon, Nayan Saxena, Naana Obeng-Marnu, Tobin South, Cole Hunter, et al. 2024. Consent in crisis: the rapid decline of the AI data commons. *arXiv preprint arXiv:2407.14933* (2024).

[36] Dhruv Mehrotra and Tim Marchman. 2024. Perplexity is a Bullshit Machine. *Wired* (2024). https://www.wired.com/story/perplexity-is-a-bullshit-machine/.

[37] Joe Miller. 2013. Jonathon Fletcher: forgotten father of the search engine. *BBC News* (2013). https://www.bbc.com/news/technology-23945326.

[38] Kien Pham, Aécio Santos, and Juliana Freire. 2016. Understanding website behavior based on user agent. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1053–1056.

[39] Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. *arXiv preprint arXiv:2104.06683* (2021).

[40] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems* 35 (2022), 25278–25294.

[41] Yang Sun, Ziming Zhuang, and C Lee Giles. 2007. A large-scale study of robots. txt. In *Proceedings of the 16th international conference on World Wide Web*. 1123–1124.

[42] Jamba Team, Barak Lenz, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, et al. 2024. Jamba-1.5: Hybrid Transformer-Mamba Models at Scale. *arXiv preprint arXiv:2408.12570* (2024).

[43] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[44] Udger. 2025. YisouSpider Details. https://udger.com/resources/ua-list/bot-detail?bot=YisouSpider.

[45] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. 2003. CAPTCHA: Using hard AI problems for security. In *International conference on the theory and applications of cryptographic techniques*. Springer, 294–311.

[46] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. 2008. recaptcha: Human-based character recognition via web security measures. *Science* 321, 5895 (2008), 1465–1468.

[47] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. 2018. Yet another text captcha solver: A generative adversarial network based approach. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 332–348.

A STATISTICAL TESTS FOR CHANGE IN BOT COMPLIANCE

In Table 9, we report z-scores and p-values for the observed changes in compliance from the most common 26 bots in our `robots.txt` experiments. As is evident, many bots exhibit statistically significant changes in behavior in response to the various `robots.txt` versions we deploy, indicating some level of awareness of the `robots.txt` protocol and/or willingness to comply.

Bot Name	**Crawl delay**		**Endpoint**		**Disallow**	
	z-score	p-value	z-score	p-value	z-score	p-value
AcademicBotRTU	-0.74	4.59e-01	0.18	8.59e-01	0.48	6.29e-01
Amazonbot	0.61	5.44e-01	10.71	0.00e+00	11.43	0.00e+00
Apache-httpclient	0.42	6.73e-01	0.99	3.21e-01	N/A	N/A
Applebot	-0.45	6.49e-01	-0.20	8.45e-01	-0.04	9.66e-01
Axios	-0.77	4.39e-01	N/A	N/A	N/A	N/A
BrightEdge	2.12	3.42e-02	1.21	2.25e-01	N/A	N/A
Bytespider	-1.96	5.00e-02	-5.04	4.74e-07	-1.30	1.93e-01
ChatGPT-User	-2.07	3.86e-02	-0.18	8.60e-01	20.62	0.00e+00
Claudebot	0.44	6.62e-01	8.95	0.00e+00	8.24	2.22e-16
Coccoc	0.27	7.84e-01	2.67	7.51e-03	2.34	1.91e-02
DataForSEOBot	2.41	1.62e-02	6.35	2.14e-10	-1.11	2.68e-01
Dotbot	-0.17	8.63e-01	4.28	1.85e-05	4.66	3.12e-06
Facebookexternalhit	0.95	3.43e-01	2.49	1.27e-02	3.98	6.97e-05
Go-http-client	14.05	0.00e+00	19.24	0.00e+00	5.23	1.72e-07
GPTBot	11.54	0.00e+00	7.43	1.09e-13	24.20	0.00e+00
HeadlessChrome	-3.15	1.63e-03	-5.15	2.67e-07	0.87	3.83e-01
Iframely	0.26	7.97e-01	-2.49	1.26e-02	N/A	N/A
MicrosoftPreview	-1.00	3.15e-01	N/A	N/A	N/A	N/A
PerplexityBot	-0.36	7.20e-01	6.86	7.01e-12	-0.90	3.69e-01
PetalBot	0.58	5.64e-01	-0.27	7.86e-01	2.81	4.94e-03
Python-requests	11.46	0.00e+00	5.04	4.64e-07	1.83	6.71e-02
SemanticScholarBot	10.39	0.00e+00	6.86	6.99e-12	7.36	1.81e-13
SemrushBot	0.91	3.62e-01	9.90	0.00e+00	10.18	0.00e+00
SeznamBot	-0.49	6.26e-01	1.65	9.91e-02	2.83	4.65e-03
SkypeUriPreview	0.95	3.44e-01	1.66	9.61e-02	N/A	N/A
Slack-imgProxy	0.35	7.24e-01	N/A	N/A	N/A	N/A
Yandex.com/bots	-1.77	7.64e-02	-0.81	4.21e-01	-0.74	4.58e-01

Table 9. Statistical significance of changes in compliance in response to each `robots.txt` version. *We report z-scores and p-values for the* 26 *most common bots across all experiments.*