OWLER: PRELIMINARY RESULTS FOR BUILDING A COLLABORATIVE OPEN WEB CRAWLER

M. Dinzinger*, S. Zerhoudi,
M. Al-Maamari, M. Istaiti,
J. Mitrović, M. Granitzer,
University of Passau, Passau, Germany

Abstract

In the rapidly evolving digital landscape, the need for efficient and effective web crawling mechanisms is more crucial than ever. Web crawlers are instrumental in discovering and indexing new content, and their role in areas such as search engine optimization, data mining, and web archiving is indispensable. However, current distributed crawling frameworks face significant challenges in terms of topic-based content discovery and categorization. This paper presents novel extensions to the StormCrawler and URLFrontier frameworks to enhance web crawling efficiency and relevance. The OWler, a derivative of the StormCrawler, is extended with classification functionalities, including topic identification and thus enabling the production of topic-specific WARC files using multiple writing streams. Concurrently, the URLFrontier framework is extended to enable web crawlers to retrieve URLs based on topic interests. To put it in a nutshell, these extensions allow us to build a highly distributed network of heterogeneous web crawlers, which are nevertheless efficiently collaborating over a shared crawl space.

INTRODUCTION

As the digital landscape continues to evolve, the need for efficient web data acquisition becomes increasingly prominent. We currently experience an urgent demand for data which empowers researchers and businesses in Europe and around the world. However, tapping the web as a resource is technically challenging and requires considerable upfront investment in infrastructure and the recruitment of highly skilled professionals. In this light, the consortium of the OpenWebSearch.eu project has pooled their resources and expertise with the aim of democratizing access to web information, currently controlled by a small group of gatekeepers, and thereby unlocking the potential for a broader community of European innovators.

Open Web Index

The common objective of the project participants is to establish an Open Web Index (OWI) and foster an ecosystem around it [7]. Central to this ambition is the advocacy for an open search engine market and the offering of a genuine choice to users. Furthermore, the Web Index provides an avenue for the development of diverse web-data products which go beyond the scope of web search and target e.g., research in the flourishing AI domain. The creation and maintenance of the OWI adhere to Open Data principles and legal compliance, and require a decentralized structure among multiple European institutions.

Web crawling

An important part of this effort is the collection of web content through crawling. A web crawler is a tool designed for systematically downloading an extensive number of webpages. Its specific software architecture varies considerably depending on the intended use of the software [2].

Our crawling system is characterized by a collaborative crawling strategy. We have opted for a distributed architecture which consists of multiple heterogeneous worker nodes in different remote computing sites. The crawling nodes communicate with a central, controlling component, the URL Frontier. This design diverges from other state-of-the-art distributed crawling systems, such as Apache Nutch [11] or BUbiNG [2], which are built upon a single software solution and threfore assume a single-machine setup or structural uniformity among the cooperating agents.

In this context, we have developed the OWler (Open Web Crawler), an incremental and distributed web crawler. It is highly customizable, yet primarily tailored for extensive data acquisition with the purpose of building a Web Index. The OWler collects webpages that align with its scope of interest and documents its fetch activities in the WARC file format. It is further augmented by our integration of a topic classification model for more specific content categorization. Furthermore, the usage of the *URL Frontier* framework as a central component enables seamless collaboration among multiple geographically distributed crawlers [15]. It maintains the known and to-be-fetched URLs, and divides the crawl space among the worker nodes based on their scope of interest.

In the remaining sections of this paper, we will present the OWler and delve into our approach to web crawling. Section 2 provides an overview of related work in this field and Section 3 describes the existing frameworks which underpin our approach. Our main contributions are outlined in Section 4, which details the enhancements made to the OWler. Section 5 evaluates the applicability of our crawling system in a real-world setup and Section 6 concludes the paper, with an outlook to future directions of our work.

RELATED WORK

The domain of web crawling dates back to the origins of the world wide web itself. Concurrent to the rise of the mod-

* michael.dinzinger@uni-passau.de

ern internet in early 1990s, tools and techniques for efficient web search became necessary. So throughout the following years, search engine operators and reseachers worked on software tools for the efficient traversal of the web. Due to a wide range of research endeavors, crawling systems have continously improved on its four main quality criteria: coverage and freshness, politeness and robustness.

Thereby, a number of technical challenges have been tackled, such as the near duplication detection of web documents [5, 14], the de-duplication of URLs [1] and the queue-based scheduling mechanism for webpages [18]. This scheduling mechanism is often implemented as *Frontier* and prioritizes URLs depending on the webpage quality, while ensuring politeness towards web servers.

Several notable web crawlers have been developed over the years. *Mercator*, described in Najork et al. (2002), was one of the first commercial open-source crawlers developed by IBM that targeted high-performance and introduced the "frontier" data structure [18]. *Heritrix* and the open-source crawler *Apache Nutch* were further early web crawlers that were extensively used and studied [11, 17]. *IRLbot* was a pioneering effort in scaling web crawling to handle billions of webpages on a single-machine setup [13]. More recently, the *BUbiNG* crawler was developed by the Laboratory of Web Algorithmics as a next-generation dataset crawler, with a public repository available for research use [2].

In the last 25 years, two web protocols have made a significant impact on the ethicality and efficiency of crawlers and, thus, have become the informal standard among webmasters and search engine operators. The *Robots Exclusion Protocol (REP)* [10] encourages content owners to state access rules for non-human visitors in a "robots.txt" file, which is placed at the website's root directory. However, the REP has no direct legal relevance [19] and is only intended to restrict disproportinate server traffic caused by robotic accesses [21]. Nevertheless, it is nowadays the most important mean for ensuring politeness in web crawling and scraping activities. At this point, we want to remark that in the future an extension to the REP or a similar technical solution might be necessary to impose fine-grained and legally binding restrictions, such as licensing information, on publicly available web content.

The *Sitemap Protocol* originates from the idea of making web servers more crawler-friendly which was initially discussed by Brandman et al in 1999 [3] and jointly implemented by Google, Yahoo and Microsoft as a common initiative in 2006. Hereby, website admins expose a list of URLs along with additional metadata called Sitemaps. This simple monitoring mechanism helps to discover new pages earlier and has a positive effect on the coverage and freshness of Web Indices [20].

A further direction of research concerns the *Focused crawling*, in which the crawler means to discover and traverse only a specific part of the web. The term was coined by Chakrabarti et al in 1999 [4] and, since then, focused crawlers have been used in a number of application cases [9, 16, 22]. In order to guide the crawler towards its specific focus, machine learning algorithms for the categorization of webpages are employed, including topic classification, spam detection, and many more.

Finally, our work also leaded us to the question of legal compliance of crawling activities. Relevant literature on this topic can be found in Schellekens (2013), Gold (2018), and Krotov (2020) [8, 12, 19]. These publications examine legal texts as well as court cases and draw the picture of a non-uniform and rather uncertain legal landscape.

EXISTING FRAMEWORKS

This section describes the technical details on the software setup of the Open Web Crawler.

StormCrawler

The OWler [1], a core component of our project, is a derivative of StormCrawler [2], a widely adopted and mature open-source web crawler. This Java-based software framework is both lightweight and scalable, underpinned by a distribution layer based on Apache Storm [3]. The StormCrawler is designed to handle multiple fetcher threads to download webpages in parallel. Nevertheless, it respect crawler ethics, including the Robots Exclusion Protocol, and applies a politeness delay. The decision to build upon the StormCrawler was motivated by the following three reasons.

Firstly, thanks to the underlying Apache Storm platform, the crawler overcomes the limitations of single-machine systems. Consequently, the StormCrawler is able to obtain high performance despite the usage of commodity hardware. It therefore differentiates from many state-of-the-art crawlers such as Heritrix [17] and BUbiNG [2] which were originally designed for a single-machine setup. Another advantage of the Storm distribution layer is the robust performance of the worker nodes, benefiting from consistently high CPU and network utilization. This is a marked improvement over the batch-wise processing observed in systems such as Apache Nutch, whose performance often suufers from periodic peaks in CPU and network traffic.

Secondly, the StormCrawler framework endows the OWler with a high degree of customizability. This characteristic is crucial in meeting the broad spectrum of our requirements, ranging from general-purpose discovery crawling to more targeted, task-specific dataset crawling.

Lastly, StormCrawler's open-source nature and active community provide a wealth of resources and support. This ensures that as we adapt and extend the crawler to meet our specific needs, we are backed by a network of developers who are continually improving the core software and who can offer assistance or solutions when challenges arise.

URLFrontier

The frontier is a crawler component that monitors the status of both crawled and to-be-crawled URLs. Within the

[1] `https://openwebsearch.eu/owler/`, visited 05/31/2023
[2] `http://stormcrawler.net/`, visited 05/31/2023
[3] `https://storm.apache.org/`, visited 05/31/2023

structure of a collaborative crawling system, it takes the central position in a star-shaped architecture. For our project, we have built upon the existing URLFrontier framework in the OpenSearch-based implementation [4]. This open-source software project defines an API [5] for facilitating communication between the frontier and the crawler. The StormCrawler framework natively supports the URLFrontier programming interface, and is able to retrieve and upload URLs from the frontier with the API's `GetURLs` and `PutURLs` command, respectively.

The adoption of URLFrontier was motivated by the nature of our crawling system, which is both heterogeneous and highly distributed. With crawlers located in computing sites across Europe, which can join or be removed arbitrarily, the frontier functions as the central storage of URLs and enables peer-to-peer crawling despite the large geographic distances between the crawlers. Its performance is therefore particularly crucial and should not be a bottleneck for the crawling activities.

Moreover, the participating crawlers may differ in terms of implementation, interests, and performance, while they still have to communicate to the same central software component. Consequently, leveraging an existing API, such as the one defined in the URLFrontier project, proves beneficial and allows us to accommodate these differences efficiently.

OWLER EXTENSIONS

This paper proposes a two-pronged approach to enhance the efficiency and relevance of web crawling. First, we extend the StormCrawler framework with classification functionality and, second, we modify the URLFrontier framework to enable URL retrieval based on diverse parameters such as topic, privacy policy presence, language, etc. Consequently, the frontier can partition the crawling space on a more fine-grained level, allow participating crawlers to choose a more refined crawling strategy. In the end, both extensions can be combined and leverage a collaborative crawling system, which allows crawlers to focus on its scope of interest, and divides the crawl space accordingly.

OWler StormCrawler

The StormCrawler framework is a robust and scalable tool for web crawling. However, it lacks a mechanism for the immediate content categorization of newly discovered. To address this, we propose the addition of enhanced URL classification functionalities to the framework and show our contribution for the case of topic-based URL categorization. This new mechanism for the categorization of newly discovered URLs complements the already existing functionalities for classifying fetched and parsed web content.

The classification process is driven by a machine learning model trained on an extensive corpus of web content, capable of identifying a broad range of topics or criteria with considerable accuracy. The used dataset and classification model have been developed concurrently to this work and are presented by Al-Maamari et al [submitted to OSSYM2023].

Once the fetched web content is categorized, the crawler produces WARC files that are organized based on the identified topics. This is achieved using multiple writing streams, with each stream dedicated to a specific topic. This approach not only enhances the efficiency of the crawling process but also facilitates easy retrieval and analysis of the crawled content

OWler URLFrontier

The URLFrontier framework takes the central position in our distributed system and communicates to all participating crawlers. Via the `GetURLs` command, a crawler retrieves the next URLs to fetch. The `PutURLs` command is used by the crawlers to inform the frontier whether a fetch was successful or erroneous and to propagate newly discovered URLs to the central storage. However, the existing framework does not support parameter-based URL retrieval and partitions the crawl space among the worker nodes only with the help of a simple consistent hashing algorithm.

To address these shortcomings, we suggest an extension to the URLFrontier framework that enables web crawlers to retrieve URLs based on their scope of interest. Each crawler is connected to a *Frontier service*, which takes the man-in-the-middle position between the crawler and the actual URL storage, in our case an OpenSearch index. The Frontier service is initialized with a specific interest, e.g., the set of all benign webpages in English language. Subsequently, it only retrieves the URLs from the shared index, which lays within its scope, and offers them to the crawler. Hence, the Frontier services divide the crawl space based on their different scopes. In case several services have the same focus, the crawl space is divided evenly among them. Note that the Frontier services have an important role as they serve as buffer and abstraction layer between the crawlers and the concrete backend which persists the URLs alongside with corresponding metadata. Moreover they are crucial as the crawlers express their interest not directly, but through the frontier service that they are connected to.

The extension of the URLFrontier framework goes hand in hand the OWler extension of StormCrawler. The enhanced classification functionalities within the crawler allow to enrich the URL with essential metadata, such as the topic, the language, the webpage category (like Sitemap, Privacy Policy, Terms of Use, etc). Thanks to our OWler extensions on the StormCrawler framework, these metadata parameters can be immediately computed within the crawler, as long as only the URL and not the page content is required for the categorization. Subsequently, the frontier uses the additional parameters for a more fine-grained retrieval of URLs. While previous approaches have focused solely on topic-oriented or geographically focused collaborative crawling, our goal is to partition the web considering diverse criteria, such as regions, topics, webpage categories, and many more.

[4] `https://github.com/PresearchOfficial/opensearch-frontier`, visited 05/31/2023

[5] `http://urlfrontier.net`, visited 05/31/2023

This approach markedly improves the relevance of the crawled content, as the web crawler is more likely to retrieve content aligning with its specific interests. It also enhances the efficiency of the crawling process, as the crawler spends less time crawling irrelevant content. Additionally, with the integration of blacklisted URLs and spam filtering mechanisms, we can further optimize the crawling process.

OWLER IN ACTION

This section evaluates the OWler in the wild. We therefore run an experimental crawl on about 1.37M seed URLs over the time period of twelve hours. This small evaluation wants to showcase the applicability and the benefits of the OWler's collaborative crawling strategy. The used seed URLs are a random sample from a recent CommonCrawl (CC) dump. Note that a CC dump gives no guarantee on a fair or even distribution of URLs, so the set of seed URLs could possibly be biased in any direction.

The distributed setup consists of three crawlers. To begin with, a general-purpose crawler is interested in all benign web content. The second crawler in the experimental setup has the task of creating a dataset for spam detection and is therefore interested in web content, whose URLs indicate malicious content. Thirdly, a Sitemap crawler only retrieves URLs, which are tagged with the corresponding `"isSitemap"` metadata field. The discovery of URLs through Sitemaps contributes to a high coverage and freshness in the Web Index and is a good complement to the solely hyperlink-based discovery approach of classical crawlers. All three workers are connected to a central frontier, which controls and monitors the progress. The frontier as well as the crawlers are extended with the OWler modifications, described in Section 4.

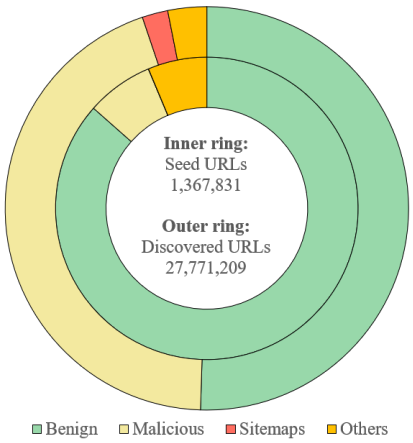

Figure 1: Increase of URLs in the experimental crawl.

The crawlers' topics of interest, `Benign`, `Malicious` and `Sitemaps`, divide the crawl space in three non-overlapping sections. Figure 1 compares the quantity and relative distribution between seed URLs and the discovered links regarding these three categories plus a fourth category with other URLs [6]. At first glance, we recognize that the proportions of the sections shift significantly. Whereas the initial mix of URLs contains a vast majority of benign webpages, an disproportionate number of links are discovered, whose URLs indicate malicious content. This observation is a hint on the importance of Focused crawling. The results in Figure 1 suggest that a set of unfocused crawlers would eventually drift off from the benign part of the web towards undesirable content, diminishing the quality of the final Web Index. It underlines once more the necessity of well-functioning classification functionalities in a crawler system, identifying malicious web content early in the process, and therefore motivates our OWler extension to the StormCrawler framework.

In summary, over 27.7M links have been discovered by the three crawlers. Thereof, 437,992 webpages have been fetched [7]. The performant general-purpose crawler has fetched the most webpages (337,224) and thereby discovered over 13.6M new URLs. The sitemap crawler has discovered the most new links per fetch, 56.6 on average. The dataset crawler profits from a high value of *topic locality* [6]. Over 86.5 % of the discovered URLs are tagged as `Malicious` and therefore lay again in its scope of interest.

With the help of the shared URL Frontier, each node is contributing to the increase of the crawl space of all its peer workers. This transfer of out-of-scope URLs leads to a mutual benefit and becomes visible in Figure 2, where the relative distribution of discovered links is denoted based on the four before-mentioned categories. The corresponding absolute numbers are listed in Table 1.

For example, this positive effect is observable in the synergy between the general-purpose and the dataset crawler. The first of these two encounters over 2.8M malicious URLs (20.3 %), which are to be avoided from its perspective, but of interest for the dataset crawler. Also in the other way around, the dataset crawler discovers benign URLs, which are of no further use for it and which are eventually transferred to its peer crawler.

Furthermore, Figure 2 attests a high degree of efficiency to Sitemap crawling in the bespoken peer-to-peer setup. The Sitemap crawler has fetched 155,276 web documents and 8.7M of the 8.9M discovered links (97.2 %) are not Sitemaps anymore, but point to HTML web content. The general-purpose and the dataset crawler have discovered Sitemaps for 160,628 domains, whereas a Sitemap crawler is natively not able to discover URLs outside the anchor domain [8] and therefore relies on the help of its peer workers in terms of discovery of new Sitemaps.

From the experiences, that we were allowed to gather so far, the StormCrawler together with the OWler extensions is

[6] The fourth category `Others` contains adult and advertisement webpages, which are classified as neither benign nor malicious.

[7] Note that due to the restricted scale of the crawl not all seed URLs have been fetched.

[8] A cross-domain reference of Sitemaps is for example possible, when the Sitemaps of a website are hosted by a third party service.

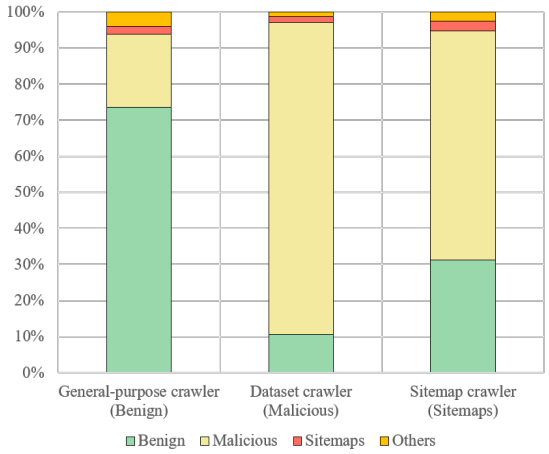

Figure 2: Distribution of discovered URLs by the three crawlers.

	General-purpose crawler	Dataset crawler	Sitemap crawler
Benign	10,331,474	536,783	2,794,105
Malicious	2,844,466	4,349,835	5,687,672
Sitemaps	289,543	82,628	246,265
Others	554,804	57,116	219,801

Table 1: Distribution of discovered URLs by the three crawlers in numbers.

able to consistently fetch up to 100 pages per second [9]. At this speed, a single crawler produces approximately 200GB of WARC files per day. These numbers refer to a StormCrawler configuration with 200 fetcher threads and four parse threads. Further experiments will allow us to work on the topology configuration, eliminate bottlenecks and improve the OWler performance further.

CONCLUSION AND FUTURE WORK

Motivated by the OpenWebSearch.eu project, our endeavors circle around the challenge of efficient and scalable web data acquisition. Our work and its preliminary results target first and foremost the OWler. The derivative of the StormCrawler has proven highly capable from a technical perspective and serves us as groundwork for customization and extensions. To begin with, we integrated a classification model in the crawling pipeline, which categorizes URLs immediately after they have been discovered. This small contribution is a first step towards a much bigger goal. We want to enrich fetched web content and discovered links with more metadata, to be able to steer the crawl with higher accuracy and ensure high quality of the desired end product, the Open Web Index.

[9] Within the experimental setup, the crawling speed was reduced to one-tenth, leading to approximately 10 pages per seconds.

The OWler extensions on the StormCrawler go hand in hand with the modifications on the URLFrontier framework. This software component keeps track of the crawl status of all discovered web resources and provides the worker nodes with next URLs to fetch. With the help of our contribution, crawlers are able to express their interests and only receive URLs within this predefined scope, which can be defined by a variety of criteria, such as regions, topics, webpage categories. This extension goes beyond the hash-based partitioning of the crawl space and targets a more generic approach to collaborative crawling.

A collaborative strategy appears to be most suiting with respect to the prevailing setup, which is highly distributed and rather heterogeneous. First tests confirm this assumption and show promising results. Several StormCrawler nodes as well as a single frontier cooperate efficiently in a shared crawl. As the observations in Section 5 suggest, the peer crawlers mutually profit from each other due to the discovery and transfer of out-of-scope URLs.

Future efforts will concern the further conceptual refinement, performance engineering and scaling of our distributed peer-to-peer OWler setup. Additionally, a more comprehensive evaluation of the crawling strategy is necessary to generate deeper, more reliable insights on its performance. This evaluation also includes the software components, which we want to extend by new features, such as the extensive parsing and processing of structured data in web documents.

Additionally, we want to set a particular focus on the topic of legal compliance in the domain of web crawling. Several research questions arise, such as (1) in which ways are copyright and licensing statements on web data objects expressed, (2) do the existing mechanisms guaratee machine-readability and are they sufficient to meet the requirements in the upcoming AI era, and (3) how can we ensure compliance to the content owners' rights, from crawling to indexing.

ACKNOWLEDGEMENT

Bundesministerium für Bildung und Forschung

This work is part of the OpenWebSearch.eu project, funded by the EU under the GA 101070014, and part of the CAROLL project, funded by the German Federal Ministry of Education and Research (BMBF) under the funding code 01|S20049.

REFERENCES

[1] Z. Bar-Yossef, I. Keidar, U. Schonfeld, *Do Not Crawl in the DUST: Different URLs with Similar Text*, ACM Trans. Web, NY, USA, 2009, pp. 380–388.

[2] P. Boldi, A. Marino, M. Santini, S. Vigna, *BUbiNG: Massive Crawling for the Masses*, ACM Trans. Web 12 (2), May 2018.

[3] O. Brandman, J. Cho, H. Garcia-Molina, N. Shivakumar, *Crawler-Friendly Web Servers*, SIGMETRICS Perform. Eval. Rev. 28 (2), NY, USA, Sep. 1999, pp. 9–14.

[4] S. Chakrabarti, M. van den Berg, B. Dom, *Focused crawling: a new approach to topic-specific Web resource discovery*, Computer Networks 31 (11-16), 1999, pp. 1623–1640.

[5] M. Charikar, *Similarity Estimation Techniques from Rounding Algorithms*, Proceedings of the 34th Annual ACM Symposium on Theory of Computing, NY, USA, 2002, pp. 380–388.

[6] C. Chung, C. Clarke, *Topic-oriented collaborative crawling*, Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM '02), NY, USA, 2002, pp. 34–42.

[7] M. Granitzer et al, *Impact and development of an Open Web Index for open web search*, Journal of the Association for Information Science and Technology, Aug. 2023.

[8] Z. Gold, M. Latonero, *Robots Welcome? Ethical and Legal Considerations for Web Crawling and Scraping*, 13 WASH. J. L. TECH. & ARTS 275, 2018.

[9] A. Juffinger, T. Neidhart, A. Weichselbraun, G. Wohlgenannt, M. Granitzer, R. Kern, A. Scharl, *Distributed Web2.0 crawling for ontology evolution*, 2nd International Conference on Digital Information Management (2), 2007, pp. 615–620.

[10] M. Koster, G. Illyes, H. Zeller, L. Sassman, *Robots Exclusion Protocol*, IETF RFC 9309, Sep. 2022.

[11] R. Khare, D. Cutting, K. Sitaker, A. Rifkin, *Nutch: A Flexible and Scalable Open-Source Web Search Engine*, 2005.

[12] V. Krotov, L. Johnson, L. Silva, *Tutorial: Legality and Ethics of Web Scraping*, Communications of the Association for Information Systems 47, 2020.

[13] H. Lee, D. Leonard, X. Wang, D. Loguinov, *IRLbot: Scaling to 6 Billion Pages and Beyond*, Proceedings of the 17th International Conference on World Wide Web, New York, NY, USA, 2008, pp. 427–436.

[14] G. S. Manku, A. Jain, A. Das Sarma, *Detecting Near-Duplicates for Web Crawling*, Proceedings of the 16th International Conference on World Wide Web, NY, USA, 2007, pp. 141–150.

[15] C. Manning, P. Raghavan, H. Schütze, *Crawling* (Chapter 20.2), In: *Introduction to Information Retrieval*, Cambridge University Press, USA, 2008.

[16] R. Meusel, P. Mika, R. Blanco, *Focused Crawling for Structured Data*, Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, NY, USA, 2014, pp. 1039–1048.

[17] G. Mohr, M. Kimpton, M. Stack, I. Ranitovic, *Introduction to Heritrix, an archival quality web crawler*, Proceedings of the 4th International Web Archiving Workshop (IWAW'04), Bath, UK, Jul. 2004.

[18] M. Najork, A. Heydon, *High-Performance Web Crawling*, Handbook of Massive Data Sets. Massive Computing (4), Springer, Boston, MA, USA, 2002.

[19] M. H. M. Schellekens, *Are internet robots adequately regulated?*, Computer Law & Security Review 29 (6), 2013, pp. 666–675.

[20] U. Schonfeld, N. Shivakumar, *Sitemaps: Above and Beyond the Crawl of Duty*, Proceedings of the 18th International Conference on World Wide Web, NY, USA, 2009, pp. 991–1000.

[21] Y. Sun, I. Councill, C. Giles, *The Ethicality of Web Crawlers*, 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, Canada, 2010, pp. 668–675.

[22] R. Zowalla, T. Wetter, D. Pfeifer, *Crawling the German Health Web: Exploratory Study and Graph Analysis*, Journal of Medical Internet Research 22 (7), Jul. 2020.