Intelligent Health Systems – From Technology to Data and Knowledge
E. Andrikopoulou et al. (Eds.)

doi:10.3233/SHTI250423

Building a Modular Health Information Crawler: Leveraging Apache Nutch for the Tala-Med Search Engine

Nanxing DENG[a], Martin BOEKER[b], and Raphael SCHEIBLE[b,1]
[a] *School of Computation, Information and Technology, Technical University of Munich, Germany*
[b] *Institute for AI and Informatics in Medicine (AIIM), TUM University Medical Center, Technical University of Munich, Germany*

Abstract. Developed in 2020, the tala-med search engine provides high-quality, evidence-based health information from trustworthy German websites while ensuring user privacy. However, it still leaves room for technical improvements. In the present work, we are replacing the Fess crawler with Apache Nutch to improve scalability and customization, as Nutch offers greater flexibility for large-scale web crawling and indexing. The new system uses Docker to integrate five services: PostgreSQL for configurations, Nutch for crawling and indexing, and ElasticSearch for search operations; a Manager orchestrates the process with custom configurations via extended Nutch-REST interface. The configuration options include domain-specific URL filters to select the crawled content. A test crawl demonstrated the system's effectiveness, processing approximately 23k websites over 65 hours. Future work will focus on deploying the crawler long-term and generating a search index for further analysis. We have published our code under the MIT license at https://gitlab.com/mri-tum/aiim/search-platform/crawler.

Keywords. Information Storage and Retrieval, Software Design, Health Information Systems, Search Engine, Online Systems, Data Systems, Web-Crawler

1. Introduction

The internet is a key source of health information, but content quality from popular general search engines varies, posing challenges for individuals with low health literacy [1-3]. General search engines often fail to meet the specific needs of healthcare professionals and diverse users, highlighting the importance of accessible and personalized health information systems [4,5]. To address this, the tala-med search engine was developed in 2020, delivering high-quality, evidence-based German health information while maintaining user privacy [6]. In contrast to the 'Sampled German Health Web' (sGHW) project [7], which employed automated filters, tala-med depends on manually curated health websites. Specht et al. [6] assessed the search engine's usability and acceptance through a study involving 802 participants. The results showed high acceptance, especially among older users and those with lower health literacy, with positive feedback on

[1] Corresponding Author: Raphael Scheible, AIIM, TUM Klinikum rechts der Isar, Ismaninger Str. 22, 81675 München, Germany; E-mail: raphael.scheible@tum.de.

the absence of ads and adjustable filters. However, some users suggested design improvements and the authors discussed the potential for technical improvements. The objective of this work is to replace the existing Fess crawler [8] with Apache Nutch [9], using its modular architecture, extensibility and built-in scalability to provide a robust solution for large-scale crawling. The new system integrates PostgreSQL for domain retrieval and ElasticSearch for indexing. To demonstrate the success of the transition, a test crawl is performed, followed by a feasibility analysis. Moreover, we have published our code.

2. Materials and Methods

2.1. Requirements and Design

The system must retrieve domains from PostgreSQL, index results in ElasticSearch, and include raw HTML for metadata tasks. It should be scalable, adaptable, and support a definable crawling schedule. Non-functionally, the system must enhance extensibility, customization, and stability, and be reliable, maintainable, and well-documented. Containerization with Docker ensures scalability and efficient management, and it can serve as a foundation for future distributed deployment. The system is modular with five services (see Figure 1): Manager (orchestrates crawling), PostgreSQL (stores configurations and history), Nutch-REST (extends Nutch's REST interface), Nutch (crawls and indexes data), and ElasticSearch (stores and provides data). These services, all containerized with Docker, communicate over a network with shared volumes.

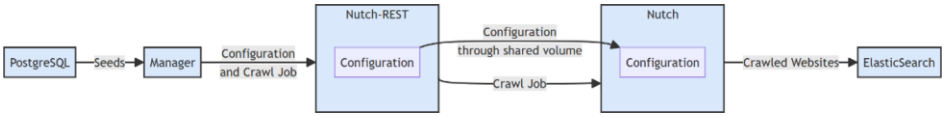

Figure 1. Flowchart of the crawling process. The Manager initiates and configures crawls using configurations stored in PostgreSQL. The Nutch-REST server configures Nutch and passes crawl jobs to Nutch which interacts with the ElasticSearch database to store the crawled data.

2.2. Implementation

PostgreSQL organizes data into seven tables, which manage seed configurations, crawl statuses, and historical records. These tables also manage seed scheduling, enabling administrators to define the frequency and timing of each crawl. The table design also includes mechanisms to log errors encountered during the crawling process, aiding in monitoring and troubleshooting.

Apache Nutch, an open-source web crawler, is designed for large-scale operations. Its flexibility stems from a plugin system, which supports custom crawling and indexing workflows. We decided to use the Nutch server instead of the crawl executable, since the server's API offers modularity in commands, and offloads the management of parallelism into the server's queue. Besides, this choice allows us to control the Nutch operations from another managing component. Crawl configurations in Nutch are managed through XML and properties files, with the primary configuration file being nutch-site.xml. Specific configuration files include index-writers.xml for index destinations and regex-urlfilter.txt for URL filtering. The system has around 1700 customizable

parameters. However, Nutch's server API lacks support for creating URL filters, demanding the development of a custom API to handle this. The regex filter helps manage which URLs are crawled by including/excluding based on patterns. This ensures efficient crawling by focusing on relevant domains and avoiding non-HTML resources, irrelevant query parameters, and infinite loops. The filter is adapted to suit the needs of each seed. The Nutch workflow begins with seed injection into the CrawlDB, which manages URL tracking. URLs are then generated into a fetch list, which is dynamically updated as new URLs are discovered. The fetcher retrieves web content from the target URLs, followed by parsing, which extracts key data like titles and links. Parsed content is indexed in ElasticSearch, facilitated by Nutch plugins that store raw HTML content [10].

Nutch-REST extends Nutch's server API, addressing limitations in managing URL filter files. Implemented as a dockerized FastAPI [11] server, it acts as a proxy, managing API calls between the Manager and Nutch. The Nutch-REST container shares a volume with Nutch, enabling file system access. It distinguishes between standard Nutch API calls and custom endpoints for managing URL filters and index configurations. The Manager uses this setup to create tailored configurations for each seed, allowing dynamic and flexible crawling processes.

Manager coordinates the entire workflow, connecting PostgreSQL, Nutch-REST, Nutch, and ElasticSearch. It relies on several libraries and tools. Our nutch-python client, a forked and customized version of the one originally developed by Chris Mattmann [12], manages interactions with the Nutch server, creating configurations for seeds and triggering processing tasks. It includes a RegexClient for handling URL filters via the custom Nutch-REST endpoint. The PostgresClient uses SQLAlchemy [13] to interact with PostgreSQL, retrieving and updating seed and filter data. It also logs crawl jobs based on Nutch responses, enabling progress tracking. Scheduled by Cron, the Manager retrieves active seeds from PostgreSQL and checks their crawl schedules. It configures and runs crawl jobs by setting up seed entries and URL filters in Nutch. Parameters like depth (levels of crawling) and topN (number of URLs to fetch) are configurable for each seed. The Manager monitors the job, logging status updates and handling errors.

2.3. Test Crawl

To evaluate the system's feasibility, a test crawl was conducted using a VM with a 4-core Intel Xeon processor, 16 GB RAM, and Ubuntu 20.04.6 LTS with Docker. Due to hardware limitations, only a subset of 17 domains from the original tala-med fetch list was used. The crawl ran for 65 hours with 40 parallel threads, and each seed was set to crawl every minute, ensuring continuous operation without downtime. The crawling parameters were set with a topN of 100 (width) and a crawl depth of 10, allowing a maximum of 1000 documents per crawl job. Custom regex URL filters were applied for each domain, designed to restrict crawling within the target site's domain and exclude non-German content. This configuration ensured the crawl was efficient and focused, tailored to the specific content needs while working within the system's resource constraints.

3. Results

The evaluation of the Nutch crawler system focuses on feasibility using a subset of seeds, with metrics logged in PostgreSQL and documents indexed in ElasticSearch. Over a 65-

hour period, the test crawl captured 22,686 documents across 36 crawl jobs per seed. Document count varied significantly depending on domain size and the strictness of URL filters. The experiment's documents per crawl job over time are depicted in Figure 2.

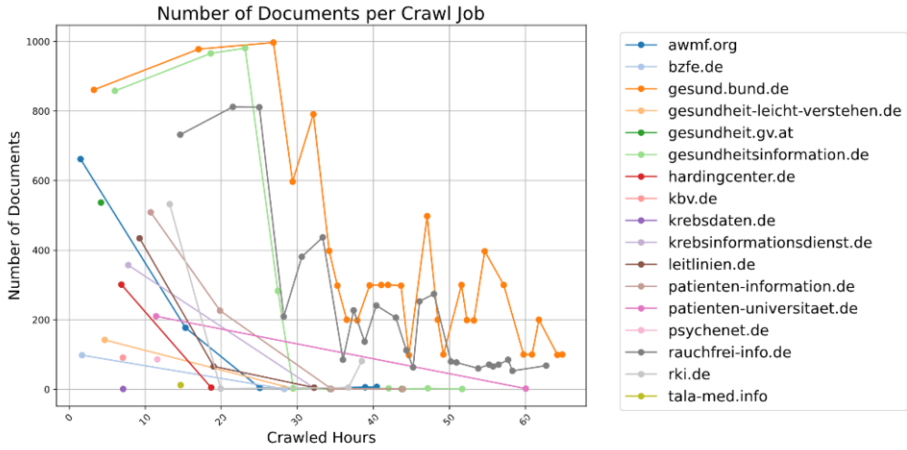

Figure 2. The number of documents per crawl decreases as the experiment progresses due to limited domain size and URL filter settings. The maximum number of documents per crawl is 1000, the product of topN 100 and depth 10.

The number of documents per crawl declined over time as available content within domains was exhausted. The maximum document count of 1,000 per crawl was determined by the configured crawl width (topN of 100) and depth (10 levels). As stricter regex filters limited the range of crawled URLs, fewer new documents were found, leading to shorter crawl durations. The experiment effectively demonstrated the system's capabilities while exposing the effects of seed configurations, especially when handling small or tightly filtered domains.

4. Discussion

While the current system faced challenges such as memory overflows due to insufficient heap size, deploying it within Kubernetes clusters, with Helm charts or Kubernetes manifests, could be explored in future work as a complementary approach to vertical scaling, enhancing resource management and supporting the efficient processing of larger datasets. Additionally, restrictions imposed by some domains' robots.txt files hindered content retrieval, which is beyond the scope of this study. Despite these issues, the system's integration potential is strong. It can connect with a frontend application, enabling graphical management of seeds via an API, improving user experience. The collected data holds value for Natural Language Processing tasks like content analysis and sentiment analysis. With extended operation, this crawler could generate an index analog to the "Sampled German Health Web" project [7], supporting analyses of health-related content in the German language, as demonstrated by prior research in the field [14].

5. Conclusions

In this work, we successfully developed and deployed a new web crawler using Apache Nutch, replacing the existing Fess crawler while meeting functional and non-functional requirements. The system retrieves domains from PostgreSQL, indexes results in Elastic-Search, and includes raw HTML content for future tasks. The test crawl, using Nutch on a subset of seeds previously used by Fess, showed the convergence of the number of documents over time towards zero as crawlable content was exhausted due to domain size and URL filters. Despite limited hardware, the Nutch crawler indexed 22,686 documents over 65 hours with 36 crawl jobs per seed. This confirms the system's functionality and sets the foundation for further evaluation and optimization.

Acknowledgments

This work was supported by the German Ministry for Education and Research grant number 01ZZ1804A, 01KX2121 and 01ZZ2304A.

References

[1] Schaeffer D, Vogt D, Berens EM, Hurrelmann K. Gesundheitskompetenz der Bev¨olkerung in Deutschland: Ergebnisbericht [report]. Universit¨at Bielefeld, Fakult¨at f¨ur Gesundheitswissenschaften. 2016. Available from: https://pub.uni-bielefeld.de/record/2908111.

[2] Schaeffer D, Berens EM, Vogt D. Health literacy in the German population: results of a representative survey. Deutsches¨Arzteblatt International. 2017;114(4):53. Publisher: Deutscher Arzte-Verlag GmbH.

[3] Schaeffer D, Berens EM, Vogt D, Gille S, Griese L, Klinger J, et al. Health literacy in Germany: Findings of a representative follow-up survey. Deutsches Aerzteblatt International. 2021;118(43):723. Publisher: Deutscher Arzte-Verlag GmbH.

[4] Kritz M, Gschwandtner M, Stefanov V, Hanbury A, Samwald M. Utilization and Perceived Problems of Online Medical Resources and Search Tools Among Different Groups of European Physicians. J Med Internet Res. 2013 Jun;15(6):e122. Available from: http://www.jmir.org/2013/6/e122/.

[5] Zhang Y. Beyond quality and accessibility: Source selection in consumer health information searching. Journal of the Association for Information Science and Technology. 2014 May;65(5):911-27. Available from: https://asistdl.onlinelibrary.wiley.com/doi/10.1002/asi.23023.

[6] Specht L, Scheible R, Boeker M, Farin-Glattacker E, Kampel N, Schm¨olz M, et al. Acceptance and usability of an independent, non-commercial search engine for medical information: a cross-sectional questionnaire study and user-behavior tracking (Preprint). JMIR Human Factors; 2024.

[7] Zowalla R, Wetter T, Pfeifer D. Crawling the German Health Web: Exploratory Study and Graph Analysis. J Med Internet Res. 2020 Jul;22(7):e17853.

[8] codelibs/fess: Fess is very powerful and easily deployable Enterprise Search Server.;. Accessed: 2024-08-23. Available from: https://github.com/codelibs/fess.

[9] apache/nutch. The Apache Software Foundation; 2024. Accessed: 2024-08-23. Available from: https://github.com/apache/nutch

[10] nutch/src/plugin at rda-crawl · b-cube/nutch · GitHub;. Accessed: 2024-08-23. Available from: https://github.com/b-cube/nutch/tree/rda-crawl/src/plugin.

[11] Ram´ırez S. FastAPI; 2024. Accessed: 2024-08-23. Available from: https://github.com/fastapi/fastapi.

[12] Deng N. dengamusic/nutch-python; 2024. Accessed: 2024-08-23. Available from: https://github.com/dengamusic/nutch-python.

[13] SQLAlchemy;. Accessed: 2024-08-23. Available from: https://www.sqlalchemy.org.

[14] Zowalla R, Pfeifer D, Wetter T. Readability and topics of the German Health Web: Exploratory study and text analysis. PLOS ONE. 2023 Feb;18(2):e0281582. Publisher: Public Library of Science.