DOI: https://doi.org/10.15276/hait.07.2024.27
UDC 004.94

Constructing a website graph using the crawling procedure

Ivan O. Dolotov[1)]
ORCID: https://orcid.org/0000-0002-4643-3464; dolotov_i@fpm.dnu.edu.ua
Natalia A. Guk[1)]
ORCID: https://orcid.org/0000-0001-7937-1039; huk_n@fpm.dnu.edu.ua. Scopus Author ID: 54791066900
[1)] Oles Honchar Dnipro National University, 72, Science Ave. Dnipro, 49010, Ukraine

ABSTRACT

The paper presents an approach to analyzing website structures. The objective is to develop an automated data collection procedure (crawling process) that systematically traverses a website and constructs a web graph represented as either lists of vertices and edges or an adjacency matrix, enabling subsequent analysis of structural connections between its elements. An unclear website structure can hinder user navigation and slow down the indexing process for search engines. Consequently, the development of automatic structure analysis methods is a relevant task. Existing information collection procedures for websites are deficient in providing comprehensive dataset and lack configuration options for setting data collection parameters. Considering that modern websites often have dynamic structures, which leads to variations in URL composition, this work enhances the approach to automating website structure data collection, accounting for dynamic pages and the specific features of their URL structure. The research method involves analyzing both internal and external links on webpages to understand the interconnections between different parts of a site. The quality of the structure is evaluated by calculating metric characteristics of the generated web graph, including diameter, density, clustering coefficient, and others. In this work a crawling procedure and algorithm were developed based on a breadth-first traversal of the graph. Software was developed to implement the crawling procedure and analyze the collected data, utilizing Python libraries such as requests, BeautifulSoup4, and networkx. Web graphs of several websites of various types and topics were constructed. The web graph representation allowed to explore the website's structural properties. Graphs were created to demonstrate the dependence between the average density of web graphs and the number of vertices, the average graph formation time and the number of vertices, and the average modularity coefficient relative to the average clustering coefficient. It was found that websites with well-defined thematic structures exhibit higher modularity and clustering coefficients. The practical significance of this work lies in its potential applications for optimizing website structures and developing new tools for data analysis.

Keywords: Graph; website; web graph; crawling; breadth-first search; clustering; modularity; transitivity; metric

***For citation*: Dolotov I. O., Guk N. A. "Constructing a website graph using the crawling procedure". *Herald of Advanced Information Technology*. 2024; Vol.7 No.4: 384–392. DOI: https://doi.org/10.15276/hait.07.2024.27**

INTRODUTION

In the modern world, each second saved during an internet search holds significant value. The vast amount of data available online makes information retrieval a challenging task for users, as it is impossible to manually review every page related to a specific topic. Consequently, search engine design requires increasingly advanced algorithms to rank results based on user relevance and present the most pertinent information on the first page. The exponential growth of online data necessitates continuous advancements in search engines to maintain the relevance of search results.

Not all available information is useful. Search engine techniques can become ineffective or produce low-quality results if the information they retrieve is not engaging to users, especially when malicious actors exploit popular keywords to manipulate traffic to their websites.

The challenges lie in ensuring the relevance, resilience, and scalability of search engines, as well as addressing the issue of retrieving web pages that may not contain specific keywords but are highly relevant to a given query.

The complex and constantly evolving nature of the modern internet demands innovative approaches to analysis. Modeling websites to represent their structure and interconnections enables a more detailed understanding of their underlying architecture, which is crucial for designing effective search algorithms. This approach remains an active and critical area of research in order to enhance search engine performance.

LITERATURE REVIEW

Modern websites are complex information systems that require efficient means of data representation and organization to ensure their structure, accessibility, and interoperability with other systems. One of the common methods of representing websites is ontology [1]. It is a

formalized structure that allows organizing website's data using concepts and their connections, providing understanding and reuse of knowledge about the subject area. Ontological models transform websites from regular page collections into structured, interpretable resources accessible to both humans and machines.

Paper [2] provides a detailed description of the application of the OWL ontology language for the formal representation of website semantics. OWL enables websites to achieve a well-structured data organization based on standardized ontologies. This facilitates semantic compatibility across various systems and resources. Such an approach is especially relevant for large information portals, such as scientific databases or media platforms, where data accuracy and completeness are critical.

Another model for representing a website is as a web graph [3]. In works [4, 5] the site's structure is considered as a directed graph, where everything is present in a hierarchy. By considering a website as a directed graph, web pages can be viewed as vertices, and hyperlinks as edges. This model provides tools for analyzing complex interconnections within websites and plays a significant role in many processes, including search engine optimization.

Both ontologies and web graphs have significant advantages for representing and analyzing websites. They allow transitioning from a static representation of information to a dynamic and semantically rich environment where data can be viewed and used for automated processing.

By providing a rich and structured representation of knowledge, ontologies contribute significantly to semantic information processing. This enhances both machine understanding and the ability to bridge the gap with human cognition. Web graphs, on the other hand, provide a global overview of the website's structure, allowing for efficient analysis of links between pages and making decisions based on this data. Ontological models are more detailed, but as a result, they are much more complex and time-consuming to analyze than graph models. Therefore, web graphs are often preferred for operational analysis of the site's structure and interconnections.

In order for a website to be effective, it is necessary to constantly collect and analyze data about its visitors. This information allows understanding how users interact with the site, which pages are the most popular, and which ones need improvement. Through analysis, it is possible to optimize the site's structure, increase its loading speed, make it more user-friendly, find and fix security issues. To represent the website's structure and analyze the cnnections between pages, a graph model will be used in this work [6, 18].

Works [7, 8], [9, 10], [11] use web crawling as a method of collecting data from a website, which involves automatically traversing web pages and saving the necessary information in the form of a web graph. This is a tool for search engines and websites to collect data, analyze relationships, and keep websites up-to-date.

There are two main approaches to implementing a crawler: global and local [12]. The global approach involves scanning a large network of sites, which requires significant computational resources and is primarily used by search engines. The local approach, on the other hand, focuses on a limited part of the web space, such as a specific site or group of sites. The choice between the global and local approach depends on the scale of the task, available resources, and specific goals of the developer.

The following types of crawlers are distinguished, depending on the strategies embedded in them: general purpose, adaptive, breadth-first crawler, hidden, parallel and distributed [13]. The type of crawler implementation depends on the type of task to be solved.

An adaptive crawler [14] can analyze the structure of websites using machine learning, highlighting the most important information and optimizing the data collection process, but its training takes a long time. A parallel crawler [14, 15] uses multiple processes simultaneously to collect and process information, making it effective for analyzing large sites or group of sites, but it requires a lot of resources. A breadth-first crawler [16] is simple to implement, does not require powerful machines for processing, but is effective only for analyzing small websites. The search operation of the crawler in [4] is described as a traversal of a directed graph. The crawler starts from primal page and follows the hyperlinks found on it, moving from one page to another. It analyzes each visited page, looking for new links, updates the current web graph, and continues its path.

Creating custom data collection tools allows you to adapt the research process to specific tasks and requirements, ensuring more accurate and relevant results. Custom crawler development provides more opportunities for customization and optimization of the data collection process for specific tasks, which is important for ensuring high-quality research results.

OBJECTIVE AND RESEARCH TASKS

The objective of this research is to develop an automated procedure for collecting data of the structure of a website (a crawling procedure). This procedure will enable site traversal and facilitate the construction of a web graph, represented as vertex and edge lists or an adjacency matrix, for subsequent structural analysis.

To achieve this objective, the following research tasks were identified:

1. Analyze existing mathematical models for representing websites and select a model that provides a suitable basis for structural analysis of the website.
2. Develop a procedure and algorithm for collecting information on external and internal links on web pages. This is necessary to construct a web graph and to understand the connections between different parts of the site and its overall structure.
3. Investigate the structural characteristics of modern websites to incorporate these features into the design of the website crawling algorithm.
4. Develop software that implements the proposed approach to web graph construction.
5. Identify the appropriate metric characteristics for analyzing website structure quality.
6. Apply the developed approach to analyze existing websites.

PROBLEM STATEMENT

The task is to develop a procedure and algorithm for automatically traversing HTML documents on a website to identify internal connections between its pages and to construct a web graph based on the collected data. When constructing the graph, ıt is crucial to consider the dynamic structure of websites and to develop methods for processing the collected data, including URL normalization, duplicate detection, and filtering of irrelevant information.

The work involves developing a software tool, using the chosen programming language and relevant libraries, to implement the crawling algorithm and analyze the collected information. The web graphs will be represented as lists of vertices and edges for further analysis. For the constructed web graphs, metric characteristics will be calculated to enable structural analysis. Finally, the developed tool will be applied to analyze the structure of existing websites of varying themes and sizes.

MATERIALS AND METHODS OF RESEARCH

1. Mathematical model of the website

A website is a collection of web pages that are interconnected and unified under a single domain name. These pages displayed as individual documents and may include text documents, images, videos, interactive elements, and more, comprising the main content that users see when they visit the site. Each page has a unique URL and contains hyperlinks to certain other pages within the website, establishing connections that facilitate user navigation.

We define a hypertext model of a website H as a set consisting of two sets: $H = \{P, L\}$ where $P = \{p_1, p_2, ..., p_n\}$ is the set of website pages; $L = \{1 \mid \exists p_1, p_2 \in l(p_1, p_2)\}$ is the set of hyperlinks between pages.

The structure of the hypertext model of a website corresponds to a mathematical model in the form of a directed unweighted graph $G = (V, E)$, where $V = P$, $E = L$. In the constructed graph V is the set of vertices, the elements of which correspond to pages of the site, E is the set of weighted edges of the graph, the elements of which correspond to hyperlinks between pages.

By constructing a web graph, we can analyze the structural connections between pages and evaluate the user experience in terms of navigation clarity.

2. Procedure and algorithm for constructing a web graph

To construct a web graph, a crawling procedure based on a breadth-first traversal of the hypertext structure is developed [5]. This approach ensures that all vertices are visited if the graph is finite and connected. Visiting all pages is essential when comprehensive information about the site is required. The algorithm systematically explores all adjacent vertices at the current depth level before moving to the next, enabling efficient discovery of new branches in the graph and finding all accessible vertices. The breadth-first traversal uses a "queue" data structure to store visited vertices, which makes the algorithm simple to implement in modern programming languages and efficient in memory usage.

The search for links located on web pages involves a systematic scan of the HTML code of each page to identify all hyperlinks leading to other web pages [17]. These links may lead to pages

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

within the same site (internal links) or to pages on other sites (external links). The parser iterates through the text strings of the page's markup, searching for <a> tags containing the href attribute. The value of the href attribute holds the URL of the linked page. For website structure analysis, only internal links are relevant. Therefore, the procedure is designed to compare the domains of the found pages with the domain of the initial page to ensure they match.

The web graph construction process is cyclical, as the algorithm may revisit a previously accessed page or encounter one already queued for processing. Pages undergo a duplicate check, and termination conditions are verified to avoid infinite loops. Several termination conditions exist: time limits, page count limits, queue depletion, reaching a specific depth, cycle detection, or encountering a special tag. The choice of termination conditions depends on the specific task and the resources allocated for web graph construction.

The web graph construction process, as described in [9], comprises three main stages: data acquisition, data extraction, and data transformation. During the data acquisition stage of the crawling procedure, data on the website is obtained through HTTP requests sent to the server. The HTML code of the page is received in response to the request. In the data extraction stage, programming language built-in tools and libraries are applied to parse the HTML code of the received page and search for <a> tags containing the href attribute. External links are ignored in this process.

Information regarding pages and their interlinkages must be converted into a structured format suitable for storing and representing the web graph. During the data transformation stage, the software generates a file to store the constructed web graph, for example, in the .graphml format.

The proposed approach to constructing the crawling procedure has been implemented in the following algorithm:

Algorithm

Initialization Step: Identify the page P_1, designated as the homepage. Initialize data structures for information storage: queue Q for unprocessed pages P_i and label array $m[P_i]$ to mark visited pages.

1. Insert P_1 into the queue Q, and set $m[P_1]=1$.
2. Read the HTML code of the first page P_i in the queue Q. Search for all links from page P_i . Convert links to absolute URLs, normalize them, check for duplicates, and record unique links in the hyperlink set L. Add all pages P_j linked with P_i to Q if $m[P_j] = 0$. Assign a label $m[P_j] = 1$ upon adding them to the queue. Once all pages linked with P_i are added to Q, add P_i to the set of pages P and remove it from Q. Return to the beginning of this step.

If no new pages or links are found, the algorithm terminates.

Data Transformation Stage: Export the resulting sets P and L in .graphml format and as text files: "pages.txt" with a numbered list of vertices and "edges.txt" with a list of edges.

EXPERIMENT SETUP

1. Features of the software implementation of building a crawler

Since modern websites contain both static and dynamic pages, there is a distinction between base and relative URLs. For further processing, as outlined in [17], it is necessary to convert hyperlinks into absolute URLs. This conversion involves combining the relative URL with the base URL, removing extraneous characters (such as double slashes and spaces), and verifying the syntactic correctness of the resulting absolute URL.

The URL conversion process can be represented by the function:

F(URL_relative,URL_base)=URL_absolute.

In the Python programming implementation, the urljoin function from the urllib.parse library is used to combine relative and base URLs into an absolute URL format.

During the URL normalization stage, it is essential to convert different representations of the same address into a canonical form. This process involves removing unnecessary parameters, fragments, and other URL variations that do not affect the content. Normalizing URLs in this way helps eliminate data duplication, enhances the efficiency of information retrieval, and facilitates the subsequent processing of collected data.

The crawler's software implementation is built as a console application in Python, utilizing various libraries for working with URLs and graphs, including:

- requests: for sending HTTP requests;
- BeautifulSoup4: for HTML parsing;
- networkx: for graph processing;
- urllib: for URL normalization;
- matplotlib: for visualizing the resulting graph.

The website scan involved utilizing the requests library to retrieve HTML content from each page. Subsequently, the BeautifulSoup4 library was employed to extract hyperlinks from the retrieved

HTML. This iterative process continued until the entire website was traversed, resulting in the generation of a comprehensive web graph. Data storage was managed using the networkx library, which enables saving the graph as a .graphml file for further analysis with other software. The data of the web graph was saved in two files: “pages.txt”, containing a numbered list of vertices (website pages), and “edges.txt”, containing a list of graph edges (hyperlinks). From the edge and vertex list representation of the graph, an additional representation was constructed as an adjacency matrix A_{ij}.

To calculate the metric characteristics of the web graphs, such as the average vertex degree, number of strongly connected components, and others, tools from the networkx library were used.

Since most online stores have dynamically generated pages, URL addresses can vary depending on selected product parameters. To simplify processing, only the primary part of the URL, without dynamic variations, was added to the graph. For example, the "Насіння країни" website (https://semena-dnepr.org.ua/) has 12,634 pages, but after normalization and conversion URLs to absolute form, the web graph contains only 2,449 pages.

2. Analysis of the obtained results

The proposed website representation model, the crawler construction procedure, and the developed software were applied to construct and analyze web graphs of existing websites available on the Internet. These included the official website of DNU (http://dnu.dp.ua/), the Faculty of Applied Mathematics website (http://fpm.dnu.dp.ua/), the Faculty of Psychology website (http://fpso.dp.ua/), and the "Насіння країни" online store (http://semena-dnepr.org.ua/).

Table provides a comprehensive overview of the metric characteristics of the analyzed websites. To evaluate the structure and properties of these graphs, the following metrics were used: average vertex degree, number of strongly connected components, graph density, modularity coefficient, clustering coefficient, and transitivity coefficient.

The average vertex degree indicates the average number of links leading from one page to others, which provides insight into how well the pages on a website are interconnected. The number of strongly connected components defines the number of groups of pages where any page can be reached from any other by following hyperlinks. Graph density reflects how densely all pages on the site are connected relative to the maximum possible number of links. A higher density indicates a greater number of interconnections between pages.

The modularity coefficient characterizes the presence of distinct clusters within the graph. A high modularity coefficient indicates that the graph comprises several groups of vertices that are strongly connected internally but weakly connected to other groups.

The calculation of the modularity coefficient is calculated as follows:

$$Q = \frac{1}{|E|}\sum_{ij}(A_{ij} - \frac{e_i^{out} e_j^{in}}{|E|})\delta(v_i, v_j),$$

where A_{ij} is an element of the adjacency matrix of the web graph, $|E|$is the number of edges in graph G, e_i^{out}, e_j^{in} represent the number of outgoing edges from vertex v_i and incoming edges to vertex v_j. By utilizing the concept of modularity, it is possible to identify thematic groups of pages that belong to a similar topic.

To assess the strength of connectivity between vertices, the clustering coefficient is used.

This coefficient is calculated as follows:

$$C = \frac{1}{|V|}\sum_i \frac{2N_{tv_i}}{\max(e_i^{out}, e_j^{in}) * (\max(e_i^{out}, e_j^{in}) - 1)},$$

where $|V|$ represents the number of vertices in graph G, and N_{tv_i} is the number of triangles containing vertex v_i This coefficient can be applied to identify thematic clusters and evaluate the extent of connectivity between pages within a single topic.

The transitivity coefficient indicates the probability that two vertices connected to the same vertex will also share a connection with each other.

Transitivity is calculated as follows:

$$T = \frac{3N_t}{N_{triad}},$$

where N_t represents the number of triangles, and N_{triad} is the number of triads. The transitivity measure helps reveal strong interconnections between pages and assess the overall density of the website.

An analysis of these metric values can provide insights into the website’s structure, the organization of information, and the ease of user navigation between different pages. This analysis can indicate the level of connectivity across different thematic sections of the site, detect duplicated content, and facilitate comparisons between various websites.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

Table. **Metric characteristics of the generated web graphs**

Metric characteristics	Websites			
	dnu.dp.ua	fpm.dnu.dp.ua	fpso.dp.ua	semena-dnepr.org.ua
Number of vertices	10854	725	91	2449
Number of edges	1285779	23768	2135	319841
Average vertex degree	108.21	42.99	20.31	46.07
Number of strongly connected components	106	19	3	8
Graph density	0.006	0.0889	0.2232	0.059
Modularity coefficient	0.168	0.189	0.194	0.219
Clustering coefficient	0.64	0.77	0.796	0.833
Transitivity coefficient	0.823	0.802	0.838	0.922

Sourse: **compilated by authors**

The analysis of the obtained data indicates that the "Насіння країни" website has the highest values of modularity, clustering coefficient, and transitivity: $Q \approx 0.219$, $C \approx 0.822$, $T \approx 0.922$. This indicates the presence of thematic clusters within the site's structure. Online stores are typically designed with a clear thematic separation of content, which generally leads to higher metrics values for this type of website. In contrast, the informational sites dnu.dp.ua, fpm.dnu.dp.ua, and fpso.dp.ua have lower values for these metrics, with a tendency for these values to decrease as the site size increases.

The Faculty of Psychology website has the smallest number of strongly connected components (3) and the highest density (0.2232). This is primarily due to the smaller size of the site, which makes it more user-friendly in terms of navigation. However, despite the large number of pages and links, the "Насіння країни" site has only eight strongly connected components, indicating a limited number of subgraphs from which it is impossible to return to the homepage.

The DNU website is the largest among all the analyzed sites. An analysis of its metric characteristics shows that website size significantly impacts its structural quality. Due to the high number of links, the boundaries of clusters become less defined, and the large number of pages considerably reduces density. An increase in the number of links leads to a higher vertex degree, resulting in cluster overlap and making it more challenging to clearly assign vertices to specific clusters. Nevertheless, considering the extensive structure of the DNU website, its metrics exceed the average values for sites of similar size. For example, $Q \approx 0.168$, while the average modularity for sites with an equivalent clustering coefficient is $Q \approx 0.165$.

The analysis and optimization of such large websites without clearly defined clusters is a complex task. However, the evaluated metric characteristics provide useful recommendations for interlinking pages and for the overall re-engineering of the site.

To evaluate the structure of the selected websites in comparison with others, graphs were constructed showing the dependence between the average density of web graphs and the number of vertices (Fig. 1), the average time taken to form a graph and the number of vertices (Fig. 2), the average modularity coefficient and the clustering coefficient (Fig. 3).

An analysis of these dependencies shows that as the number of pages increases, density decreases, while the time required constructing the web graph increases. The average modularity coefficient rises with an increase of the clustering coefficient. The construction time for the "Насіння країни" web graph was significantly longer than for other sites with a similar number of vertices, indicating a higher number of links. However, its density and modularity coefficient are higher than average, indicating a stronger potential for precise clustering within the site.

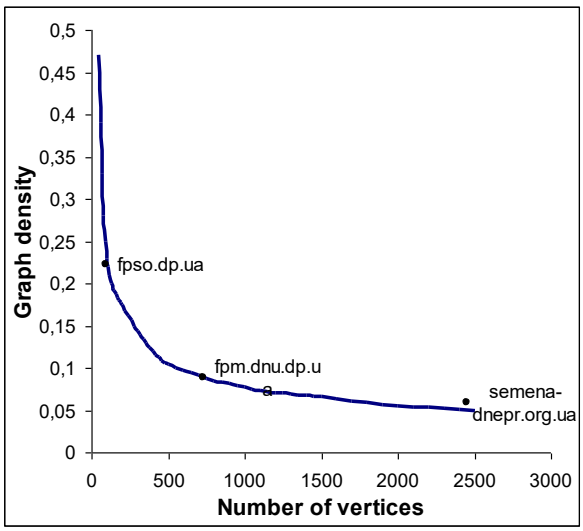


Fig. 1. **Dependence of graph density on number of vertices**

Sourse: **compiled by authors**

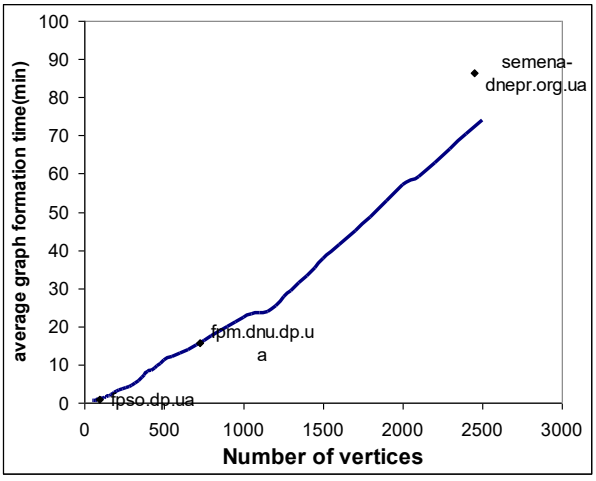


Fig. 2. **Dependence of the average time taken to form a graph on the number of vertices**

Sourse: **compiled by authors**

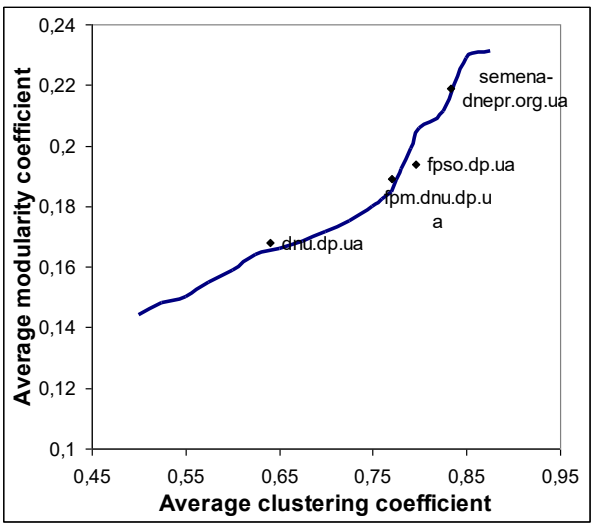


Fig. 3. **Dependence of the average modularity coefficient and the clustering coefficient**

Sourse: **compiled by authors**

The research was conducted across different types of websites. Online stores, such as the "Насіння країни" site, generally exhibit higher metrics than informational sites, so dividing websites into separate categories would provide more precise analysis.

The results obtained provide a foundation for further research on website structures. Future studies could examine the connections within subgraphs in more detail, analyze specific pages for the presence of meaningful hyperlinks or associations with clusters distinct from the primary theme, and investigate the presence of cycles within the graph structure.

CONCLUSIONS

This work presents an approach for gathering information about website structure. A model representing the website as an unweighted directed graph was used. A procedure for constructing a web graph was developed, utilizing the breadth-first search algorithm. A URL normalization procedure was implemented. The software was designed to accommodate the dynamic page construction of modern websites and was developed using Python, with specialized libraries for graphs and web data. The proposed approach was applied to construct web graphs for existing sites, which were saved in text files as a list of vertices, a list of edges, and in .graphml format for further processing. The selected sites were analyzed using various metric characteristics, including the average degree of vertices, the number of strongly connected components, graph density, modularity coefficient, clustering coefficient, and transitivity coefficient. The study indicated that modularity and clustering coefficients can be used as quantitative indicators of website structure. It was found that websites with a well-defined thematic structure tend to have higher values for these characteristics. Graphs were created showing the dependence between the average density of web graphs and the number of vertices, the average time to form a graph and the number of vertices, and the average modularity coefficient relative to the clustering coefficient. This data enabled a comparative analysis of website structure quality and average quality metrics for similar sites. The results demonstrate that representing a website as a graph is an effective tool for examining website structure and investigating its unique features. This research represents an important step toward the development of tools for automated analysis and optimization of large websites.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

REFERENCES

1. Oluchukwu, U. E., Sylvanus, O. A., Ifeoma, M. A. O. & Chukwuemeka, M. O. "Semantic web ontology technology and Its Impact on E-Learning". *American Journal of Embedded Systems and Applications,* 2021; 8 (2): 9–11. DOI: https://doi.org/10.11648/j.ajesa.20210802.11.

2. Kaung, M. H. "Web ontology language analysis (owl) and semantic web technology". *Auditorium.* 2017; 4 (16).

3. Liakos, P., Papakonstantinopoulou, K., Ntoulas, A. & Delis, A. "Rapid detection of local communities in graph streams". *IEEE Transactions on Knowledge and Data Engineering*. 2022; 34 (5): 2376–2377, https://www.scopus.com/inward/record.url?eid=2-s2.0-0035048198&partnerID=40&md5=b7684a6f28ac21a3e17929d3073a7878. DOI: https://doi.org/10.1109/TKDE.2020.3012608.

4. Casas, P., Wassermann, S., Wehner, N., Seufert, M. & Hossfeld, T. "Not all web pages are born the same content tailored learning for web QoE Inference". *IEEE International Symposium on Measurements & Networking (M&N).* 2022, https://www.scopus.com/inward/record.url?eid=2-s2.0-85140925138&partnerID=40&md5=fdc27389c27f7f74e62789dd5b5404a2. DOI: https://doi.org/10.1109/mn55117.2022.9887781.

5. Zheng, S., Dmitriev, P. & Giles, C. L. "Graph-based seed selection for webscale crawlers", *Proceedings of the 18th Conference on Information and Knowledge Management*. 2009, https://www.scopus.com/inward/record.url?eid=2-s2.0-74249122055&partnerID=40&md5=724a911529b69d88ce1bb5e5c83bbbef.
DOI: https://doi.org/10.1145/1645953.1646277.

6. Claude, F. & Navarro, G. "A fast and compact web graph representation". *International Symposium on String Processing and Information Retrieval*. 2007. p. 122–126. DOI: https://doi.org/10.1007/978-3-540-75530-2_11.

7. Chatterjee, S. & Nath, A. "Auto-Explore the Web – Web Crawler". *Article in International Journal of Innovative Research in Computer and Communication Engineering*. 2017. DOI: https://doi.org/10.15680/IJIRCCE.2017. 0504006.

8. Amudha, S. "Web crawler for mining web data". *International Research Journal of Engineering and Technology*. 2017; 4 (2): 131–134. ISSN: 2395-0072.

9. Sahu, M. B. & Bharne, S. "A survey on various kinds of web crawlers and intelligent crawler", *International Journal of Scientific Engineering and Applied Science*. 2016; 2 (3).

10. Shkapenyuk, V. & Suel, T. "Design and implementation of a high-performance distributed web crawler". *In Proceedings of the 18th International Conference on Data Engineering*. 2002. p. 357–368. DOI: https://doi.org/10.1109/ICDE.2002.994750.

11. Shrivastava, V. "A methodical study of web crawler". *Journal of Engineering Research and Application*. 2018; 8 (11): 1–5. DOI: https://doi.org/10.9790/9622-0811010108.

12. Aiello, W., Chung, F. & Lu, L. "A random graph model for massive graphs". *The 32nd Annual ACM Symposium on Theory of Computing*. 2000. p. 171–180, https://www.scopus.com/inward/record.url?eid=2-s2.0-0033633051&partnerID=40&md5=901263fc0e1ad9cb51e7ac451fc1eede.
DOI: https://doi.org/10.1145/335305.335326.

13. Zhao, L., Yin, Z., Yu, K., Tang, X., Xu, L., Guo, Z. & Nehra, P. "A fuzzy logic based solution for network traffic problems in migrating parallel crawlers". *International Journal of Advanced Computer Science and Applications*. 2023; 14 (2). DOI: https://doi.org/10.14569/IJACSA.2023.0140252.

14. Mehyadin, A. E., Abdulrahman, L. M., Ahmed, S. H. & Qashi, R. "Distributed fundamentals based conducting the web crawling approaches and types (focused, incremental, distributed, parallel, hidden web, form focused and breadth first) crawlers". *Journal of Smart Internet of Things.* 2023; 2023 (2): 10–32. DOI: https://doi.org/10.2478/jsiot-2022-0002.

15. Cheok, S. M., Hoi, L. M., Tang, S. K. & Tse, R. "Crawling parallel data for bilingual corpus using hybrid crawling architecture". *Procedia Computer Science*. 2022, https://www.scopus.com/inward/record.url?eid=2-s2.0-84964022157&partnerID=40&md5=3562db1bbd4258fc7f1a258a73fd42db
DOI: https://doi.org/10.1016/j.procs.2021.12.218.

16. Sujaini, H., Perwitasari, A. & Januardi, T. "Sistem pembelajaran algoritma best first search, breadth first search & depth first search". *Jurnal Teknik Indonesia*. 2023; 2 (2). DOI: https://doi.org/10.58860/jti.v2i2.15 (Online) https://jti.rivierapublishing.id/index.php/rp.

17. Guk, N. A., Dykhanov, S. & Matiushchenko, O. "Algorithm for building a website model". *Bulletin of V. N. Karazin Kharkiv National University Series "Mathematical Modeling. Information Technology. Automated Control Systems"*. 2020; 47: 2–3. DOI: https://doi.org/10.26565/2304-6201-2020-47-03.

18. Dolotov, I. & Guk, N. A. “Clustering of a weighted webgraph with the usage of modularity”. *Applied Mathematics and Mathematical Modeling Issues*. 2023; 23: 25–32. DOI: https://doi.org/10.15421/322305.

Conflicts of Interest: The authors declare that they have no conflict of interest regarding this study, including financial, personal, authorship or other, which could influence the research and its results presented in this article

Received 12.09.2024
Received after revision 30.10.2024
Accepted 14.11.2024

DOI: https://doi.org/10.15276/hait.07.2024.27
УДК 004.94

Побудова графа вебсайту з використанням процедури краулінгу

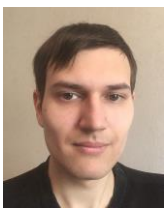
Долотов Іван Олександрович[1)]
ORCID: https://orcid.org/0000-0002-4643-3464; dolotov_i@fpm.dnu.edu.ua.
Гук Наталія Анатоліївна[1)]
ORCID: https://orcid.org/0000-0001-7937-1039; huk_n@fpm.dnu.edu.ua. Scopus Author ID: 54791066900
[1)] Дніпровський національний університет ім. Олеся Гончара, проспект Науки, 72. Дніпро, Україна

АНОТАЦІЯ

Розглянуто підхід до аналізу структури вебсайту. Мета роботи полягає у розробці процедури автоматичного збору даних про структуру вебсайту (процедури краулінгу), за допомогою якої здійснюється обхід сайту та будується вебграф у вигляді списків вершин та ребер або матриці суміжності, для подальшого вивчення структури через аналіз зв’язків між його елементами. Незрозуміла структура вебсайту призводить до погіршення навігації сайтом для користувача та уповільнення індексації сайту пошуковими машинами, тому розробка процедур автоматичного аналізу структури є актуальною задачею. Відомі процедури збору інформації про сайт не забезпечують можливість отримання повного набору даних та не мають налаштувань для визначення параметрів збору інформації. Враховуючи, що сучасні вебсайти мають динамічну структуру, яка призводить до відмінностей у записі URL-адрес, у роботі вдосконалюється підхід до автоматизації збору інформації про структуру сайту з врахуванням наявності динамічних сторінок та особливостей побудови їхніх URL-адрес. Методом дослідження є вивчення зовнішніх та внутрішніх посилань на вебсторінках для розуміння зв’язків між окремими частинами сайту, оцінювання якості структури через визначення метричних характеристик побудованого вебграфа, зокрема діаметру, щільності, коефіцієнту кластеризації тощо. В роботі розроблено процедуру та алгоритм краулінгу, що спираються на метод обходу графа в ширину. Для реалізації процедури краулінгу та аналізу отриманих даних розроблено програмне забезпечення із використанням бібліотек Python (requests, BeautifulSoup4, networkx). Побудовано вебграфи кількох вебсайтів різного спрямування та тематики. Зображення сайту у вигляді вебграфа дозволило дослідити його структуру. Побудовано графіки залежності середньої щільності вебграфів від кількості вершин, середнього часу формування графа від кількості вершин та середнього коефіцієнту модулярності від коефіцієнту кластеризації. Встановлено, що вебсайти з чітко вираженою тематичною структурою мають більш високі значення коефіцієнтів модулярності та кластеризації. Практична значущість роботи полягає в тому, що отримані результати можуть бути використані для оптимізації структури сайтів та розробки нових інструментів для аналізу даних.

Ключові слова: граф; вебсайт; вебграф; краулінг; обхід в ширину; кластеризація; модулярність; транзитивність; метрика

ABOUT THE AUTHORS



Ivan O. Dolotov - Postgraduate student, Faculty of Applied Mathematics. Oles Honchar Dnipro National University, 72, Science Ave. Dnipro, 49010, Ukraine
ORCID: https://orcid.org/0000-0002-4643-3464; dolotov_i@fpm.dnu.edu.ua.
***Research field*:** Webgraph modeling; clustering analysis

Долотов Іван Олександрович - аспірант, факультет Прикладної математики. Дніпровський національний університет імені Олеся Гончара , пр. Науки, 72. Дніпро, 49010, Україна

Natalia A. Guk - Doctor of Physical and Mathematical Sciences, Professor, Faculty of Applied Mathematics. Oles Honchar Dnipro National University, 72, Science Ave. Dnipro, 49010, Ukraine
ORCID: https://orcid.org/0000-0001-7937-1039; huk_n@fpm.dnu.edu.ua. Scopus Author ID: 54791066900
Research field: Machine Learning; intelligent information technologies; mechanics

Гук Наталія Анатоліївна - доктор фізико-математичних наук, професор, факультет Прикладної математики. Дніпровський національний університет імені Олеся Гончара , пр. Науки, 72. Дніпро, 49010, Україна