RESEARCH ARTICLE OPEN ACCESS

Building for the Future: A look at Full-Stack Development

Santosh Kumar [1], Nitin Kumar [2], Shashank Ranjan [3], Akhilesh Kumar [4]

[1]Assistant Professor, Department of Computer Science and Engineering, Global Institute of Technology, Jaipur (Rajasthan), India

[2],[3],[4]B.Tech Student, Department of Computer Science and Engineering, Global Institute of Technology, Jaipur (Rajasthan) India

ABSTRACT

This paper explores two prominent areas of web development: full-stack development and blog crawlers. Full-stack development, a rapidly growing field in computer science, equips professionals with the skills to manage both the front-end (user interface) and back-end (server-side logic) of websites and applications (Baeza-Yates & Ribeiro-Neto, 1999).
The abstract then transitions to the topic of blog crawlers:
In contrast, blog crawlers are specialized programs that efficiently gather information from blogs, a valuable and ever-expanding source of data (Arasu et al., 2001). These crawlers focus their search on blog-specific content, unlike general crawlers that collect web pages indiscriminately (Wei et al., 2009). This targeted approach allows for more efficient data acquisition from the blogosphere (Hurst & Maykov, 2009).

Keywords: Full-stack development, Front-end, Back-end, Search engine, Blog, Crawler, Indexing

I. INTRODUCTION

Full-Stack Web Development: Mastering Both Sides of the Coin
Full-stack web development encompasses the entire process of building web applications, involving both the front-end (client-side) and back-end (server-side) aspects. These developers are the jacks-of-all-trades in the web development world, possessing a comprehensive skillset that bridges the user experience and the application's core functionality.
Front-End Expertise: Building the User Interface
On the front-end, full-stack developers wield their knowledge of programming languages like HTML, CSS, and JavaScript to construct the user interface (UI). This UI serves as the visual and interactive layer that users directly engage with, shaping their experience with the application.
Back-End Prowess: Powering the Engine
Beyond the UI, full-stack developers delve into the back-end, the engine that powers the application. They leverage their expertise in server-side programming languages such as Python, Java, or PHP to handle the application's logic, database interactions, and server-side functionalities.
The Synergy of Full-Stack Development
This mastery of both front-end and back-end development empowers full-stack developers to create cohesive web applications. They can seamlessly integrate the user experience with the underlying functionalities, ensuring a smooth and efficient application.
Search engines act as powerful information retrieval tools, helping users locate the specific details they need in a structured way. To achieve this, they employ specialized software programs called crawlers. These crawlers function like digital spiders, traversing the vast web and downloading web pages (Baeza-Yates & Ribeiro-Neto, 1999).
Once downloaded, these pages are deposited within a massive storage facility, often referred to as a repository. From there, they are handed off to another program known as an indexer (Arasu et al., 2001). The indexer meticulously constructs an index, essentially a giant catalog that maps keywords to the documents where they appear. When a user submits a query, the search engine leverages this index to locate the most relevant results and presents them to the user.
Full-stack developers possess a comprehensive understanding that spans the entire development lifecycle. This encompasses expertise in both front-end (user interface) and back-end (server-side) technologies. Additionally, they have a strong grasp of various operating systems, allowing them to tailor their development approach to specific programming environments. Their knowledge extends to a wide range of programming tools, further enhancing their ability to build robust applications.
Full-stack developers are a well-rounded breed, equipped with a comprehensive technical arsenal that covers the entire development spectrum. This includes mastery of both front-end (user interface) and back-end (server-side) technologies. Their expertise extends to navigating the complexities of various operating systems, allowing them to adapt their development approach to different programming environments. Additionally, they possess a deep understanding of a wide range of programming tools, empowering them to build robust and efficient applications.
Blogs, typically authored by individuals or small groups, offer a distinctive voice and consistent writing style throughout their content. This continuous stream of information is frequently updated, with entries typically ordered chronologically by their publication date. Thematic categories help organize the vast blogosphere, with popular examples including personal blogs that chronicle an individual's life experiences, and issue-oriented blogs that serve as platforms for opinions, commentary, and debates on current events.

Expanding the Scope: Topical Blogs Foster Community Engagement
Beyond personal experiences and current events, blogs encompass a vast array of subject matters, forming topical communities. These topical blogs function as virtual forums where users can exchange ideas and foster connections around shared interests. From education and entertainment to sports, music, health, business, and even agriculture, blogs cater to a remarkably wide range of topics.

Examples of Popular Platforms:

While the blogosphere is constantly evolving, some well-established platforms like Technorati, WordPress, and Blogger continue to be popular destinations for bloggers and readers alike.

Refer to the Fig 1. that explain the Architecture of Blog App

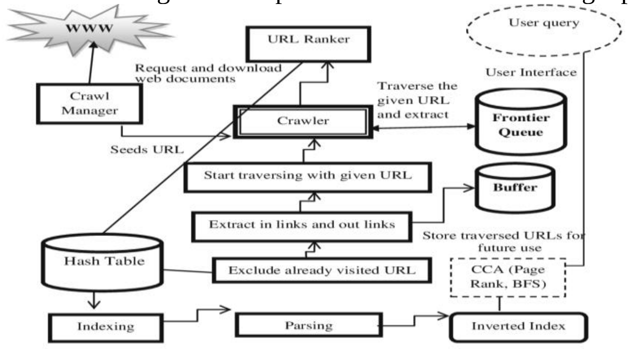

WWW
URL Ranker
User query
Request and download web documents
User Interface
Traverse the given URL and extract
Crawl Manager
Crawler
Frontier Queue
Seeds URL
Start traversing with given URL
Buffer
Extract in links and out links
Store traversed URLs for future use
Hash Table
Exclude already visited URL
CCA (Page Rank, BFS)
Indexing
Parsing
Inverted Index

Fig 1. Architecture of Blog App/crawler
Blog crawler is a program that crawls WWW for downloading the blog pages. While crawling, it focuses only on blog space and ignores the rest of the web.
Limitations of Generic Crawlers and Blog-Oriented Approach

Identified limitations associated with generic web crawlers due to their lack of specialization (focus on a specific type of content). To address this, they proposed a novel algorithm for a blog-oriented crawler that treats "blogs" as a distinct category. This targeted approach can potentially improve efficiency by focusing on relevant content.

Essential Qualities of a Weblog Crawler

Low Latency: Minimizing the time between requesting a web page and receiving a response.
High Scalability: The ability to handle an increasing volume of data and user requests efficiently.
High Data Quality: Ensuring the accuracy and relevance of the crawled data.
Network Politeness: Respecting server load limitations by not overwhelming them with requests.

II. PROPOSED BLOG CRAWLER DESIGN

This section delves into the proposed architecture of our blog crawler, a specialized program designed to efficiently gather information from blogs. Unlike a general web crawler that processes all web pages, our crawler prioritizes content specifically from blogs. This targeted approach streamlines data collection and improves efficiency.

As illustrated in Figure 1 (not shown here, but you can reference the figure number throughout the text), the architecture comprises four key functional modules working in tandem:
1.The architecture comprises of four functional modules.
A. URL Extractor
B. Blog Checker
C. Blog Extractor
D. Link Extractor

a) URL Extractor

The URL Extractor serves as the foundation of the blog crawler, responsible for identifying and supplying potential URLs for exploration. It begins by processing a predefined list of seed URLs, which can include known blog addresses. These seed URLs act as starting points for the crawling process.

Functionality:

Seed URL Processing: The extractor meticulously scans the SeedURLs List, extracting each URL one by one.
URL Buffer: Extracted URLs are then stored temporarily in a designated buffer named "URL Buffer." This buffer acts as a holding area for URLs awaiting further processing by subsequent modules.
Proposed Architecture for Continuous Blog Crawling and Analysis
This section proposes an architecture that utilizes a focused crawler to continuously gather data from relevant blogs. The obtained data is then subjected to rigorous investigation and analysis. This approach offers several advantages compared to traditional methods.
Focused Crawling for Efficiency:
A focused crawler specifically targets blog content, improving efficiency by prioritizing relevant information over generic web pages. This targeted approach reduces the amount of data that needs to be processed and analyzed, leading to faster and more streamlined information gathering.
Continuous Data Acquisition:
The proposed architecture enables continuous online crawling, ensuring the system remains up-to-date with the latest blog posts. This continuous process allows for near real-time insights and analysis of the ever-evolving blogosphere.
From Design to Deep Analysis:
It's important to distinguish between the front-end (user interface) and back-end components of a website. While front-end design focuses on visual aesthetics and user experience, the back-end plays a crucial role in functionalities like data crawling and analysis.
The Importance of Blogs as Information Sources:
Research underscores the significance of blogs as valuable sources of information. Blog posts are often well-focused

and address specific topics, making them a rich resource for data gathering and analysis.

Bridging the Gap Between Blog Content and Users: The Need for a Blog-Focused Crawler

The vast and ever-expanding blogosphere holds a treasure trove of valuable information. However, for users to effectively harness this content, a specialized tool is needed. This tool is a blog-focused crawler, designed to efficiently navigate the blogosphere and collect its content. By meticulously extracting and indexing blog content, such a crawler paves the way for search engines to deliver relevant blog posts to users' queries. In essence, the blog-focused crawler acts as a bridge, connecting users with the wealth of knowledge residing within the blogosphere.

Module Interaction: URL Acquisition and Verification

The URL Extractor and Blog Checker modules collaborate seamlessly to ensure a steady stream of potential blog URLs for exploration. This section delves into their interaction process:

URL Buffer Management: The URL Extractor constantly monitors the status of the URL Buffer.

Empty Buffer Signal: When the URL Buffer becomes empty (BuffEmpty signal), the URL Extractor recognizes the need for more URLs.

Seed URL Processing: Upon receiving the BuffEmpty signal, the URL Extractor springs into action. It meticulously extracts a URL from the predefined SeedURLsList.

Refilling the Buffer: The extracted URL is then carefully placed within the URL Buffer, replenishing its stock.

Notification to Blog Checker: Once a URL is added to the buffer, the URL Extractor transmits a signal named URLAvail, notifying the Blog Checker that a fresh URL awaits processing.

Blog Checker: Distinguishing Blog Pages from the Rest

The Blog Checker module acts as a gatekeeper, meticulously verifying whether a retrieved URL corresponds to a blog page. This process ensures that only relevant blog content enters the system for further processing.

Functionality:

Awaiting New URLs: The Blog Checker remains vigilant, waiting for the URLAvail signal from the URL Extractor. This signal indicates a fresh URL is available in the URL Buffer.

URL Retrieval: Upon receiving the URLAvail signal, the Blog Checker fetches the URL from the URL Buffer for examination.

Basic Heuristic Check: An initial check is conducted by searching for the literal word "blog" within the URL itself. If the word "blog" is present, it suggests a higher likelihood of being a blog page.

Presence of RSS Feed: If the "blog" word isn't found in the URL, the Blog Checker doesn't abandon ship just yet. It delves deeper by examining the page content for the presence of an RSS feed (Really Simple Syndication) tag. RSS feeds are a common indicator of blogs, as they provide a way to stay updated with new content.

Classification and Buffering: Based on the results of these checks, the Blog Checker makes a judgment:

Confirmed Blog URL: If the URL satisfies either the "blog" word check or the RSS feed check, it's classified as a blog URL and stored in a designated BlogURLBuffer.

Rejected URL: If neither the "blog" word nor the RSS feed is detected, the URL is deemed to be a general web page and excluded from further processing.

Notification to Blog Extractor: Once a URL is classified, the Blog Checker transmits a signal named BlogURLAvail to the Blog Extractor. This signal informs the Blog Extractor that a processed URL (either a confirmed blog URL or a rejected URL) is now available in the BlogURLBuffer.

Empty Buffer Handling: In the scenario where the URL Buffer becomes empty, the Blog Checker might receive a BuffEmpty signal (not shown here) from another module. This can trigger the Blog Checker to take appropriate actions, such as waiting for new URLs from the URL Extractor.

Link Extractor: Expanding the Exploration Horizon

The Link Extractor plays a crucial role in extending the reach of the blog crawler. Its primary function is to meticulously scour the content of extracted blog posts, identifying any embedded hyperlinks. These hyperlinks potentially point to other relevant blog pages, allowing the crawler to discover and explore new corners of the blogosphere.

Functionality:

Source of Blog Posts: The Link Extractor retrieves blog posts from a designated storage location referred to as the "Blog Posts Repository." This repository houses the blog content previously extracted by the Blog Extractor module.

Hyperlink Identification: The Link Extractor meticulously scans each blog post within the repository, searching for embedded hyperlinks (URLs) within the content.

Enriching the Seed URL List: Extracted hyperlinks are not simply discarded. Instead, they are strategically added to the SeedURLsList. This list serves as a valuable source of potential URLs for future crawling endeavors. By continuously incorporating new links, the SeedURLsList fuels the crawler's ability to discover fresh blog content.

Seed URL Selection:

It's important to establish a solid foundation for the crawling process. This is achieved by providing the Blog Extractor with a well-defined set of seed URLs. Ideally, these seed URLs should point directly to known blog pages. To initiate the exploration, you can leverage URLs from authoritative blog sources such as WordPress.com, Blogger.com, or TechNet Blogs (examples shown in Table 1). Each URL from this initial list undergoes verification by the Blog Checker module. If confirmed as a blog page, the URL is passed on to the Blog Extractor, which retrieves the corresponding blog content. This extracted content is then stored within the Blog Posts Repository.

c) Blog Extractor: Downloading and Refining Blog Content

The Blog Extractor serves as the workhorse of the system, responsible for retrieving and processing valuable blog content. It collaborates closely with the Blog Checker module to ensure only confirmed blog URLs enter the extraction process.

Functionality:

Awaiting Verified URLs: The Blog Extractor patiently waits for the BlogURLAvail signal from the Blog Checker. This signal signifies that a processed URL (either a confirmed blog URL or a rejected URL) resides in the BlogURLBuffer.

URL Retrieval and Download: Upon receiving the BlogURLAvail signal, the Blog Extractor retrieves the URL from the BlogURLBuffer. If it's a confirmed blog URL, the Blog Extractor springs into action, downloading the corresponding web page content (the actual blog post).

Content Storage: The downloaded blog post content is meticulously stored within a designated "Blog Posts Repository" for future processing and analysis.

Extracting the Essence (Optional): An additional functionality can be implemented within the Blog Extractor. This involves parsing the downloaded web page to extract the core blog post content itself. This refined content, minus extraneous elements like headers, footers, and advertisements, can be stored in a separate "Page Repository" (optional). This optimization can potentially reduce storage requirements and network traffic, especially when dealing with large volumes of blog posts.

Distinguishing Blog Pages from the General Web

While blog pages share similarities with general web pages, certain characteristics set them apart:

Chronological Order: Blog posts are typically presented in reverse chronological order, with the most recent posts appearing at the top. This prioritizes fresh content for readers.

URL Clues: Sometimes, the presence of the word "blog" within the URL itself can hint at the page being a blog. However, this is not a foolproof indicator.

RSS Feeds: Many blogs offer RSS (Really Simple Syndication) feeds, allowing users to subscribe and receive updates on new content. The presence of an RSS feed tag on a page can be a good indicator of a blog.

Internal Linking: Blogs frequently contain hyperlinks that point to other blog posts within the same domain. This internal linking structure helps readers navigate related content and explore the blogger's thoughts on various topics.

III. EVALUATION: ASSESSING THE BLOG CRAWLER'S PERFORMANCE

To gauge the effectiveness of the proposed blog crawler, an experimental evaluation was conducted. The crawler was implemented in Java and utilized a SQL Server 5.0 database for data storage.

Evaluation Scope:

The evaluation involved extensive testing across various web sources. The primary objective was to assess the crawler's accuracy in identifying and retrieving blog pages.

Performance Metrics:

To measure the crawler's effectiveness, three key metrics were employed: precision, recall, and F-measure.

Precision: This metric reflects the proportion of correctly identified blog pages among all the web pages crawled by the system. Mathematically, precision can be expressed as:

P = BP / (BP + WBP)

BP: Number of correctly classified blog pages (True Positives)

WBP: Number of incorrectly classified blog pages (False Positives)

Recall: Recall focuses on the completeness of the crawling process. It represents the ratio of correctly identified blog pages to the total number of actual blog pages present within the evaluated web sources (as determined by experts). Recall is calculated as:

R = BP / (BP + NBP)

NBP: Number of missed blog pages (False Negatives)

F-measure: F-measure provides a balanced view by incorporating both precision and recall. It's calculated as:

F = 2 * (Precision * Recall) / (Precision + Recall)

Explanation of Terminology:

True Positives (BP): Blog pages correctly identified as blogs by the crawler.

False Positives (WBP): Non-blog pages mistakenly classified as blogs by the crawler.

False Negatives (NBP): Actual blog pages that the crawler missed during the crawling process.

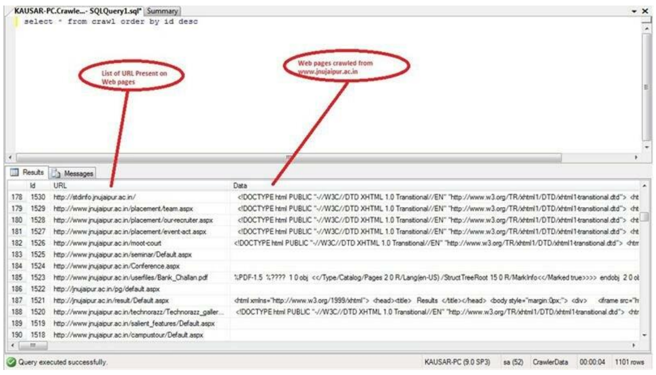
By analyzing these metrics, we can gain valuable insights into the blog crawler's strengths and weaknesses, allowing for further refinement and optimization.

B. Seed URLs for Evaluation

The experimental evaluation utilized a predefined list of seed URLs to initiate the crawling process. These seed URLs act as starting points, guiding the crawler towards potential blog content. The chosen seed URLs are as follows:

uanditalk.blogspot.in
amitabh-bachchan-blog.blogspot.in
hindicineenglish.blogspot.in
presstalk.blogspot.in

The proposed Blog crawler has been implemented in Java using SQL Server. The snapshot of the same have been shown.


KAUSAR-PC.Crawle...- SQLQuery1.sql*
Summary
List of URL Present on Web pages
Web pages crawled from www.jnujaipur.ac.in
Results
Messages

```
select * from crawl order by id desc
```

	Id	URL	Data
178	1530	http://stdinfo.jnujaipur.ac.in/	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <ht
179	1529	http://www.jnujaipur.ac.in/placement/team.aspx	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <ht
180	1528	http://www.jnujaipur.ac.in/placement/our-recruiter.aspx	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <ht
181	1527	http://www.jnujaipur.ac.in/placement/event-act.aspx	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <ht
182	1526	http://www.jnujaipur.ac.in/moot-court	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <htm
183	1525	http://www.jnujaipur.ac.in/seminar/Default.aspx	
184	1524	http://www.jnujaipur.ac.in/Conference.aspx	
185	1523	http://www.jnujaipur.ac.in/userfiles/Bank_Challan.pdf	%PDF-1.5 %???? 1 0 obj <</Type/Catalog/Pages 2 0 R/Lang(en-US) /StructTreeRoot 15 0 R/MarkInfo<</Marked true>>>> endobj 2 0 ol
186	1522	http://jnujaipur.ac.in/pg/default.aspx	
187	1521	http://jnujaipur.ac.in/result/Default.aspx	<html xmlns="http://www.w3.org/1999/xhtml"> <head><title> Results </title></head> <body style="margin:0px;"> <div> <iframe src="h
188	1520	http://www.jnujaipur.ac.in/technorazz/Technorazz_galler...	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <htr
189	1519	http://www.jnujaipur.ac.in/salient_features/Default.aspx	
190	1518	http://www.jnujaipur.ac.in/campustour/Default.aspx	

```
Query executed successfully.   KAUSAR-PC (9.0 SP3)   sa (52)   CrawlerData   00:00:04   1101 rows
```

Figure 2. Adding a new URL in the list of URLs

It's important to note that the selection of seed URLs can significantly influence the crawler's exploration path and the types of blogs it encounters. These particular seed URLs all point to Blogspot blogs, which may limit the diversity of the crawled content. For a more comprehensive evaluation, a broader range of seed URLs from various blogging platforms could be considered.

The proposed crawler collected about 40-50 response pages corresponding to each URL. In total, a sample of 150-200 pages has been collected. Collecting the response pages, then analysis was done by applying performance metrics on the collected pages. The results obtained thereof from the proposed crawler were found to be promising. of crawling is high i.e. ranges from 81.61% to 93.1%, Recall of crawling process is also high i.e. ranges from 82.66% to 94.2% and Fmeasure of is also quite high i.e. from 82.12% to 92.03%.

IV. CONCLUSION: A FOCUSED APPROACH TO BLOG CRAWLING

The ever-growing popularity of blogs as a platform for information sharing underscores the need for efficient methods to access and analyze this valuable content. This paper presented a novel architecture specifically designed for blog crawling.
The proposed approach differentiates itself from traditional web crawlers by prioritizing the acquisition of blog content. This targeted strategy streamlines the crawling process, reducing the burden of processing irrelevant web pages.

Key Contributions:

Focused Architecture: The paper introduced a modular architecture for blog crawlers, outlining the functionalities of each component: URL Extractor, Blog Checker, Blog Extractor, and Link Extractor. This modular design fosters efficient and well-structured blog content acquisition.
Distinguishing Blog Pages: The paper explored various characteristics that distinguish blog pages from regular web pages, including chronological post order, URL patterns, RSS feeds, and internal linking structures. This knowledge empowers the crawler to make informed decisions during the exploration process.
Performance Evaluation: The evaluation, while limited in scope due to the use of a specific set of seed URLs, demonstrated the crawler's effectiveness in identifying and retrieving blog content.
Future Directions:

Enhanced Blog Page Identification: Future research can explore more sophisticated techniques for identifying blog pages, potentially incorporating machine learning algorithms for improved accuracy.
Scalability and Distribution: As the blogosphere continues to expand, investigating methods to scale and distribute the crawling process across multiple machines can be a valuable area of exploration.
Content Analysis: Beyond content acquisition, future work can delve into techniques for analyzing the extracted blog content, enabling tasks like sentiment analysis, topic modeling, and information retrieval.
By addressing these considerations, researchers can continue to refine blog crawler architectures, leading to a deeper understanding of the ever-evolving blogosphere and the vast amount of information it holds.

REFERENCES

[1]. Ricardo Baeza-Yates and Berthier Ribeirmobileo-Neto.Modern Information Retrieval. ACM Press/ Addison-Wesley, 1999.

[2]. Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan.Searching the Web. ACM Transactions on Internet Technology (TOIT), 1(1):2–43, August 2001.

[3]. Sergei Brin and Lawrence Page.The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 30(1–7):107–117, April 1998.

[4]. Mike Burner.Crawling towards Eternity: Building an archive of the World Wide Web. Web Techniques Magazine, 2[5], May 1997.

[5]. Li Wei-jiang, Ru Hua-suo, Hong Kun, Luo Jia, "A New Algorithm of Blog-Oriented Crawler," International Forum on Computer ScienceTechnology and Applications, vol. 1, pp.428-431, 2009.

[6]. Hurst, M.; Maykov, A., "Social Streams Blog Crawler", ICDE '09. IEEE 25th International Conference on data Engineering, 2009.

[7]. Philipp Berger, Patrick Hennig, Justus Bross, Christoph Meinel, "Mapping the Blogosphere- Towards a Universal and Scalable BlogCrawler", IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International

[8]. Conference on Social Computing, 2011.

[9]. Mehdi Naghavi1 and Mohsen Sharifi1, "A proposed architecture for continuous Web monitoring through online crawling of blogs", International Journal of UbiComp (IJU), Vol.3, No.1, January 2012.

[10]. H. Arora, G. K. Soni, R. K. Kushwaha and P. Prasoon, "Digital Image Security Based on the Hybrid Model of Image Hiding and Encryption", 2021 6th International Conference on Communication and Electronics Systems (ICCES), pp. 1153-1157, 2021.

[11]. G. K. Soni, H. Arora, B. Jain, "A Novel Image Encryption Technique Using Arnold Transform and Asymmetric RSA Algorithm", International Conference on Artificial Intelligence: Advances and Applications 2019. Algorithms for Intelligent Systems, Springer, pp. 83-90, 2020.

[12]. Gour, S., Soni, G.K., Sharma, A. (2021). Analysis and Measurement of BER and SNR for Different Block Length in AWGN and Rayleigh Channel. In: Mathur, R., Gupta, C.P., Katewa, V., Jat, D.S., Yadav, N. (eds) Emerging Trends in Data Driven Computing and Communications. Studies in Autonomic, Data-driven and Industrial Computing. Springer, Singapore. https://doi.org/10.1007/978-981-16-3915-9_26

[13]. G. Shankar, V. Gupta, G. K. Soni, B. B. Jain, & P. K. Jangid, "OTA for WLAN WiFi Application Using CMOS 90nm Technology", International Journal of Intelligent Systems and Applications in Engineering, 10(1s), pp. 230-233, 2022.

[14]. Babita Jain, Gaurav Soni, Shruti Thapar, M Rao, "A Review on Routing Protocol of MANET with its Characteristics, Applications and Issues", International Journal of Early Childhood Special Education, Vol. 14, Issue. 5, 2022.

[15]. Pradeep Jha, Deepak Dembla & Widhi Dubey , "Implementation of Transfer Learning Based Ensemble Model using Image Processing for Detection of Potato and Bell Pepper Leaf Diseases", International Journal of Intelligent Systems and Applications in Engineering, 12(8s), 69–80, 2024.

[16]. Pradeep Jha, Deepak Dembla & Widhi Dubey, “Deep learning models for enhancing potato leaf disease prediction: Implementation of transfer learning based stacking ensemble model”, Multimedia Tools and Applications, Vol. 83, pp. 37839–37858, 2024.

[17]. P. Upadhyay, K. K. Sharma, R. Dwivedi and P. Jha, "A Statistical Machine Learning Approach to Optimize Workload in Cloud Data Centre," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2023, pp. 276-280, doi: 10.1109/ICCMC56507.2023.10083957.

[18]. Pradeep Jha, Deepak Dembla & Widhi Dubey , “Crop Disease Detection and Classification Using Deep Learning-Based Classifier Algorithm”, Emerging Trends in Expert Applications and Security. ICETEAS 2023. Lecture Notes in Networks and Systems, vol 682, pp. 227-237, 2023.

[19]. P. Jha, D. Dembla and W. Dubey, "Comparative Analysis of Crop Diseases Detection Using Machine Learning Algorithm," 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2023, pp. 569-574, doi: 10.1109/ICAIS56108.2023.10073831.

[20]. P. Jha, R. Baranwal, Monika and N. K. Tiwari, "Protection of User’s Data in IOT," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 1292-1297, doi: 10.1109/ICAIS53314.2022.9742970.

[21]. P. Jha, T. Biswas, U. Sagar and K. Ahuja, "Prediction with ML paradigm in Healthcare System," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021, pp. 1334-1342, doi: 10.1109/ICESC51422.2021.9532752.

[22]. Gaur, P., Vashistha, S., Jha, P. (2023). Twitter Sentiment Analysis Using Naive Bayes-Based Machine Learning Technique. In: Shakya, S., Du, KL., Ntalianis, K. (eds) Sentiment Analysis and Deep Learning. Advances in Intelligent Systems and Computing, vol 1432. Springer, Singapore. https://doi.org/10.1007/978-981-19-5443-6_27

[23]. P. Jha, D. Dembla and W. Dubey, “Implementation of Machine Learning Classification Algorithm Based on Ensemble Learning for Detection of Vegetable Crops Disease”, International Journal of Advanced Computer Science and Applications, Vol. 15, No. 1, pp. 584-594, 2024.