https://doi.org/10.70265/HCQX8107

OSINT TOOLS FOR REGIONAL DYNAMICS EVALUATION

Michael Dimitrov

Summary: *This study compares the performance of two web content extraction tools, one based on computer vision (EE) and the other implementing Puppeteer (3P-IO), in the context of their ability to quantify the regional security. Using a dataset of 100 webpages, the evaluation focuses on crawl time, scrape time, precision (BLEU score), and load success rate. The results reveal significant differences: 3P-IO is faster in both crawl and scrape times, while EE achieves a 100% load success rate, ensuring reliable content retrieval. Despite similar precision, EE's reliability is a key advantage. The study proposes a hybrid approach that combines 3P-IO's speed and precision with EE's reliability, activating the computer vision component only when 3P-IO's fallback strategy fails. This approach offers a balanced solution, leveraging the strengths of both tools. The study provides valuable insights for stakeholders interested in the measurability of regional security dynamics.*

Keywords: *Web Content Extraction, Regional Security, Computer Vision, Headless Browsing, Scrape Time, Precision (BLEU Score), Hybrid Approach*

INTRODUCTION

The ever-evolving security landscape demands a comprehensive approach to threat detection and mitigation. Even seemingly insignificant background trends can hold valuable insights, especially when considering regional dynamics. In this context, Open-Source Intelligence (OSINT) tools have emerged as powerful assets for security professionals seeking to gain a deeper understanding of potential risks. While many tasks cannot yet be automated and still rely on human processing, the overall data flowing through electronic information channels of interest is readily amenable to evaluation by automated solutions.

Furthermore, measuring and analyzing political processes, particularly those occurring on the international stage, has long been a complex challenge. However, advancements in AI are paving the way for new methods to overcome these hurdles and are providing venues not only for data generation but for obtaining information with a higher degree of structure and meaning. This convergence has the potential to empower nations to glean deeper insights into the ethno-demographic landscape which

remains one of the culprits for statehood erosion (Ninov & Nikolov, 2023; Vasilev & Borisova, 2023) and counter the dissemination of disinformation and propaganda in the context of "sharp power" (Atanasov, 2023), thus demonstrably altering the security environment.

This article focuses on a specific aspect of the OSINT toolkit – web content extraction. While both programs discussed herein possess advanced data processing capabilities, our primary focus will be on their core functionalities for gathering new information. Even so, exemplary results regarding the evaluation of the security dynamics in the Western Balkans will be presented. This will further enrich the text and its potential to contribute to the ongoing dialogue in relation to the best practices and limitations in the usage of machine learning for OSINT tasks.

We hypothesize that the two data extraction approaches will exhibit statistically significant differences in their performance, an anticipated outcome. The true value of this research lies in identifying where these differences are concentrated. One of the primary limitations of this study is its reliance on programs exclusively developed by the author. Other potentially effective programs (Krotov, Johnson, & Silva, 2020) remain outside of its scope. Additionally, the specificity of the targeted websites and their data might introduce some distortion into the observed results. Thus, the research implies its value only in the context of regional security. Finally, the ethical considerations surrounding OSINT tool usage (Krotov, Johnson, & Silva, 2020), a crucial aspect for real-world applications, are not explored in this article. It should be noted that the author has adhered to avoiding any negative impact on the targeted servers.

The findings presented may captivate a diverse range of audiences. Security professionals and researchers (Hayes & Cappa, 2018; Andric & Terzic, 2023) may benefit from the comparative analysis, as well as developers seeking to explore information channels with higher degree of protective measures. Overall, workstation-level automation may appeal to stakeholders in contexts beyond the one examined here.

METHODOLOGY

The comparison between the two programs is initiated with the launch of EE to extract 100 webpages pertinent to regional security. Their selection is based on the program's built-in mechanisms, ensuring sufficient diversification of resources and representing a real-world application rather than a controlled "laboratory" scenario. 3P-IO is launched second because of its capability to process a predetermined set of URLs. Following the data collection, the evaluation process is conducted based on four key parameters: crawl time, scrape time, precision (BLEU score), and load success rate.

Crawl and scrape times for both programs were measured manually using the Stopwatch class in the runtime environment. The precision of the

extracted content from both programs was evaluated using the BLEU score at the corpus level. Measurements were conducted with BleuNET, utilizing reference text manually extracted by the author to ensure maximum accuracy. This n-gram approach, while more commonly used for machine translation evaluation, is particularly suitable for our context as it offers a precise measure of "loss of information". This is especially relevant for 3P-IO, which can extract similar content from sources other than the initially targeted ones. In line with this clarification, it must be noted that the load success rate was calculated as the proportion of the initial 100 websites that were successfully loaded by both programs.

Mann-Whitney U tests were implemented using Accord.NET for the collected data across each of the four parameters to determine whether there were statistically significant differences between EE and 3P-IO. This ranking-based approach is suitable for ensuring that the comparison between the programs is meaningful. The calculations are based on the following formula:

$$U = n_1 n_2 + \frac{n_1(n_1 + 1)}{2} - R_1 \quad (1)$$

Where:

- n_1 and n_2 are the sample sizes of the two groups.
- R_1 is the sum of the ranks for the first group.

The confirmation of statistically significant differences by the Mann-Whitney U tests enables the opportunity to conduct a TOPSIS analysis. TOPSIS, which stands for Technique for Order of Preference by Similarity to Ideal Solution, is implemented in this study using C#. This provides a solid basis for concluding which program performs better and which are the reasons behind this state.

COMPUTER VISION TECHNIQUES FOR WEB CONTENT EXTRACTION

EE employs a machine learning approach designed to mimic human interactions with web browsers on a PC. To achieve this, it can search for icons, wait for browser responses, and confirm the scrolling rate to continuously maintain a "map" of the opened sites. EE leverages a set of four object detection models specifically designed to facilitate these actions. These models are trained using ML.NET (Microsoft.NET., n.d.), a free, open-source machine learning framework for the .NET platform. Although the program currently uses datasets of up to 300 images per model for training, it is important to recognize that this size is relatively small for robust object detection tasks. This choice stems from the program's focus on concept testing and the limitations inherent to a single-researcher approach compared to resource-intensive enterprise efforts. In general, larger and more diverse datasets can significantly improve the accuracy and generalizability

of machine learning models. However, the achieved performance with such a limited amount of data demonstrates the merits of the chosen approach.

Model training is conducted on a local machine, which limits the possible resolution and number of epochs. Through multiple attempts, the optimal threshold is determined and applied to all models. Their number is intentionally limited to four to avoid the precondition for availability of significant computational resources. This design decision aims to achieve a sensible balance between the visibility of "anchors" used during the operation of the program and its overall size on disk, which is currently close to 4GB.

Unlike Puppeteer, which operates at the browser level, EE acts as a human user to servers, utilizing a real browser instance complete with a complex search history and all typical attributes. Consequently, this approach bypasses common anti-scraping measures employed by services like Cloudflare. Furthermore, since the browser in question is the one commonly operated on the PC, the user may have already verified their identity on various sites, reducing the likelihood of anti-scraping measures being triggered against EE.

The program's core logic, which involves interacting with the existing browser, requires a development framework capable of interfacing with the operating system. Windows Forms Application (WFA) is a suitable choice for this purpose. While other frameworks might offer comparable capabilities for simulating user interactions, WFA's established integration with the Windows environment and its ease of use for desktop application development were significant factors in this choice. Additionally, WFA facilitates the creation of a flexible and user-friendly graphical user interface, allowing for the easy addition, removal, and editing of scraped web page content directly within the application. This approach simplifies the process of managing and refining the extracted data, making it readily accessible for further analysis or use.

EE prioritizes user comfort by incorporating a state-saving functionality which ensures continuous recording of its operation (including progress and extracted data) using JSON format upon a closure event. This feature empowers users to pause lengthy extraction tasks at any point and resume them later without losing progress. To enhance its efficiency, EE incorporates pre-extraction checks for content relevance based on title analysis. This targeted approach helps the program focus on extracting data that is significant, reducing processing time and improving overall speed. The Levenshtein distance metric is employed to identify potential inconsistencies between titles and their corresponding extracted content. To ensure title uniqueness and prevent the duplication of previously extracted content, a Jaccard index-based function is utilized for verification. It should be noted that EE is also designed to overcome issues arising from built-in server

refresh functions, ensuring seamless continuation of crawling operations. The program performs crawling and scraping concurrently, further streamlining the data extraction process.

The most sophisticated model of EE is the one employed for the challenging task of identifying the main body of textual data within a webpage. It aims to avoid the extraction of links to other pages, advertisements or any data not related to the main content. It is noteworthy that the program utilizes a universal algorithm for content scraping, which is not tailored to any specific website or data format. All four models perform well on the mean Average Precision (mAP50_95) metric, with values ranging from 0.5847 to 0.7315. mAP50_95 is commonly used in object detection tasks, representing the average precision achieved at an Intersection over Union (IoU) threshold of 50%. Higher mAP50_95 values indicate better model performance in correctly identifying relevant objects within the webpage. While this metric is crucial for the overall capability of EE, the precision of the extracted content holds greater value and will be evaluated using BLEU. The universality of the algorithm is quite impressive, as it is theoretically effective for any site design or input language.

EE showcases the significant potential of computer vision in web content extraction. By overcoming browser limitations and automating tasks directly on the user's workstation, it represents a logical advancement stemming from the recent developments in the field. However, this approach demands more computational resources compared to traditional browser-based scraping methods. Therefore, future research may focus on optimizing performance and reducing resource usage, possibly by exploring lighter-weight computer vision models or utilizing hardware acceleration techniques. Despite this drawback, EE's unique capabilities offer considerable value for researchers and developers engaged in web content extraction tasks, especially in situations where traditional scraping methods fall short.

ALTERNATIVE APPROACH: EFFICIENT WEB CONTENT EXTRACTION USING PUPPETEER

3P-IO is implemented as a Console app in C# with a focus on simplicity and efficiency. It maintains a remarkably low memory footprint, requiring only around 600 MB of disk space to operate. This makes it suitable for use on machines with limited resources. 3P-IO's efficiency is further enhanced by its reliance on primitive data types whenever possible and a technique that leverages a single parameter containing additional information. This information, embedded within the data itself using a specific format, allows the program to trigger state changes without requiring numerous variables for separate state tracking. By combining these approaches, 3P-IO minimizes

the overall amount of data that needs to be transferred between different parts of the program, thus reducing memory usage.

Separation of crawling and scraping tasks is another specificity of 3P-IO that is aimed at enhancing its overall flexibility. This allows it to distribute requests to target servers over time, reducing the risk of triggering anti-scraping measures. However, the use of Puppeteer Sharp, which creates browser instances, can easily be identified as automated activity. To mitigate this, 3P-IO employs logic to alternate between search engines and aggregators during the crawling phase. This mimics human-like browsing behavior, decreasing the possibility of detection.

The chosen approach offers several advantages beyond reducing anti-scraping risks. Firstly, the crawling stage generates a file containing navigation paths to relevant webpages. Users can manually edit or entirely rewrite the file to specify the exact addresses they want 3P-IO to process. This allows for targeted content extraction and the possibility of a manual check before the investment of significant time resources to the next phases. Additionally, such separation facilitates independent control over entry points for crawling and scraping. By running these operations from different IPs, 3P-IO can further enhance its stealth and make it more difficult to detect as an automated program.

3P-IO employs a multifaceted method to pinpoint the textual data of interest. This method incorporates two key components. First, it utilizes a collection of 45 site-specific strategies tailored to efficiently extract content from pre-defined addresses known to hold relevant information aligned with the program's goals. Second, it leverages a universal approach well-suited for handling a broader range of webpages commonly encountered during the scraping process. While 3P-IO lacks a traditional GUI, it prioritizes user transparency. Extracted data is saved in editable files containing the complete HTML structure and the identified main textual content delimited by markers highlighted with background colors that represent the methods used. This color-coding scheme empowers users to quickly assess extraction accuracy. They can easily refine the boundaries of the main textual data and readily identify areas requiring further attention.

After the extraction process, the resulting file containing the textual data is fed into the program's pseudorationalization component (utilized by both EE and 3P-IO). This component leverages machine learning models to automatically categorize the content into predefined classes. The process relies on multiple binary classifications and is focused on assignment of labels to the extracted information, making it easier to organize and analyze. Furthermore, the input is disaggregated into sub-categories based on their relevance for the particular evaluation scenario.

3P-IO goes beyond basic web scraping functionalities by offering a range of advanced features that enhance its usability and efficiency. One such

feature is a flexible scheduling option for mapping. This can be strategically aligned with the target server's specific load patterns to optimize performance. Additionally, users can leverage scheduling to automate crawls during off-peak hours. This approach not only saves time but also minimizes the potential impact on target servers during busy periods, exemplifying the ethical considerations in OSINT practices.

3P-IO's versatility extends beyond its data extraction capabilities. The program's design, particularly its low resource usage, makes it suitable for an intriguing application: a continuous content monitoring system. By configuring 3P-IO to crawl and analyze web content at regular intervals, it can be transformed into a real-time signaling mechanism. Users can define specific keywords, phrases, or patterns of interest. Whenever 3P-IO encounters these elements during its crawls, it can trigger pre-defined actions, such as sending alerts or notifications. This allows for proactive monitoring of online environments for critical information or emerging trends.

In conclusion, 3P-IO's capabilities extend far beyond its potential for OSINT operations conducted by state enterprises, highlighting the relevance and effectiveness of Puppeteer-based solutions. Its ability to efficiently extract and categorize web content makes it an asset for a wide range of applications in the private sector as well. Private companies can leverage 3P-IO for tasks like market research, competitor analysis, and brand monitoring. By continuously screening online conversations and identifying customer sentiment patterns, 3P-IO has the potential to empower companies to make informed decisions regarding product development and marketing strategies. As web content continues to be a rich source of valuable data, this program serves as a powerful tool for organizations seeking to stay informed and make data-driven decisions.

COMPARATIVE ANALYSIS OF WEB CONTENT EXTRACTION METHODS: COMPUTER VISION VS. PUPPETEER

Following the evaluation of the two web content extraction programs, we proceed with a comparative analysis of their performance. During the operation of both programs on a set of 100 websites, their results were recorded with respect to the four key parameters: crawl time, scrape time, precision that is calculated using BLEU at corpus level (BLEU: Bilingual Evaluation Understudy (Papineni, Roukos, Ward, & Zhu, 2002, pp. 311-318) implemented via BleuNET), and load success. It is important to note that the "load success" parameter is binary while the other three represent continuous data.

As mentioned above, 3P-IO employs a robust 4-stage fallback strategy to enhance content extraction success rates. It first attempts to process the targeted website directly. If unsuccessful, it leverages a popular search

engine to find alternative URLs based on the resource's title. The program then prioritizes content extraction from these retrieved URLs using Puppeteer, with attempts in both headless and non-headless modes to overcome potential anti-scraping measures. As a last resort, it employs a second search engine to identify alternative content sources. This multi-pronged approach significantly increases the program's resilience in the face of website loading failures or inaccessibility. Due to this, while 3P-IO might encounter unsuccessful loads, its implemented fallback strategy effectively mitigates the risk of zero precision by accessing websites with relevant content.

Considering the clarified relationship between load success and BLEU score, we are ready to present the empirical data. EE exhibits a wide range of crawl times, with many webpages being processed in under 20 seconds. However, there is significant variability in the data, with some values exceeding 70 seconds. This suggests that the program's behavior along this parameter might be influenced by website-specific factors, especially the distribution of relevant information. 3P-IO excels at finding new content quickly. It processes most navigations in under 12 seconds, with a substantial number taking less than 5 seconds.

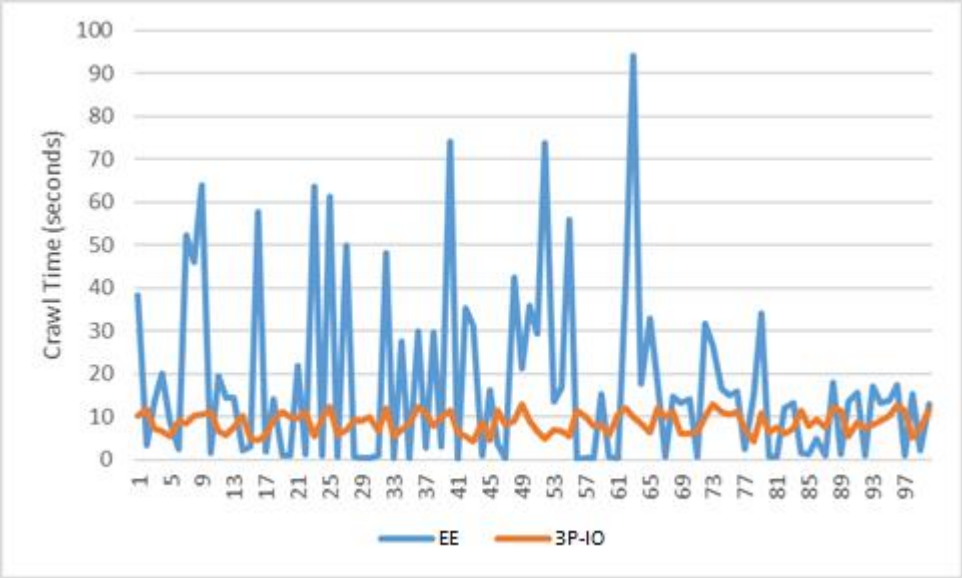


Figure 1. Comparison of Crawl Times Between EE and 3P-IO

Regarding scrape times, both EE and 3P-IO exhibited a wider range of values than initially anticipated. EE scrape times ranged from under 40 seconds to a maximum near 114 seconds, with a standard deviation of approximately 17.76 seconds. Similarly, 3P-IO scrape times showed a broader range (2.62 to 57.98 seconds) with a standard deviation of about 6.14 seconds. These variations likely arise from the complexity of the content

being scraped (presence of embedded media, length and structure of textual data).

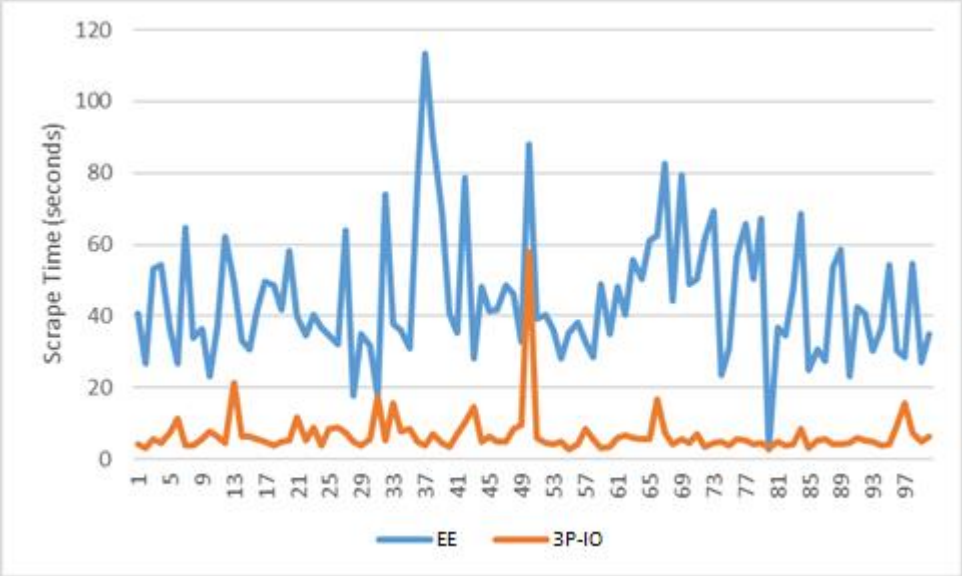


Figure 2. Comparison of Scrape Times Between EE and 3P-IO

The precision of extracted text by EE varied across tasks (BLEU scores ranging from 0.003903299 to 1), with some achieving very high n-gram overlap with the reference text. 3P-IO exhibited BLEU scores between 0 and 1. Similar to EE, 3P-IO's extracted text precision varied, but there were multiple cases of perfect matching.

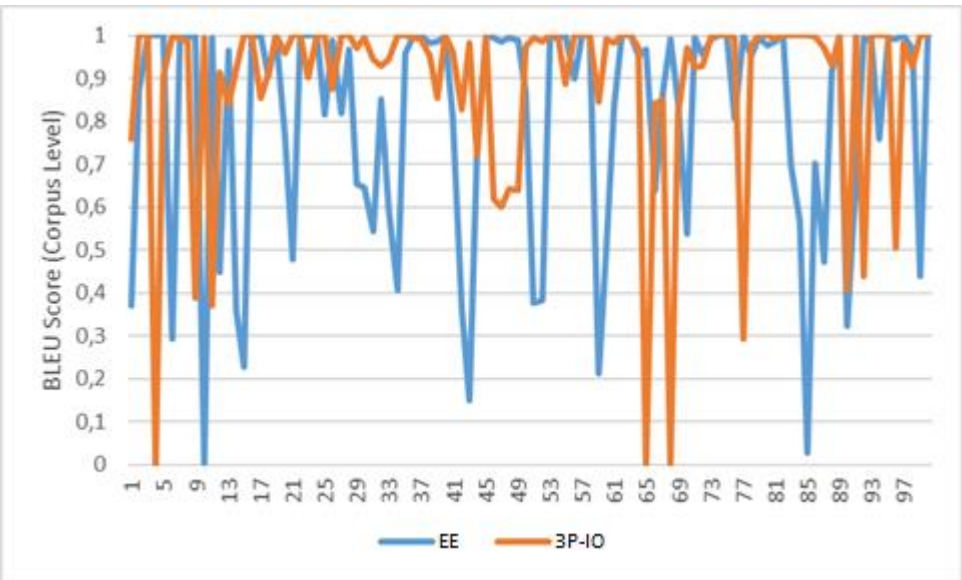


Figure 3. Precision (BLEU Score) Comparison Between EE and 3P-IO

Along the consistency of retrieval of target content, EE achieved a 100% success rate in loading webpages. 3P-IO had a 6% fail rate, indicating potential areas for improvement. This fail rate is intrinsically related to the program's Puppeteer-based nature, which, while offering robust capabilities for web content extraction, also introduces certain challenges that need to be addressed. To facilitate a comparative analysis of the programs' performance, Table 1 presents the average values across all four parameters.

Table 1. Average Performance Parameters of EE and 3P-IO

Parameter	EE	3P-IO
Crawl Time (seconds)	17.7397	8.61
Scrape time (seconds)	44.8907	6.7507
Precision (BLEU Score)	0.808910384	0.884952276
Load Success Rate (%)	100	94

Having presented the empirical data, we shall now focus on determining whether there is a statistically significant difference between the two programs. To this end, Mann-Whitney U tests are utilized. The implementation of these tests in C# using Accord.NET is straightforward.

• Crawl Time: The Mann-Whitney U test revealed a statistically significant difference in crawl times between the two programs (U = 4148, p-value = 0.0375). This p-value, less than 4%, suggests a low probability of observing such a difference by random chance.

• Scrape Time: The analysis of scrape times using the Mann-Whitney U test yielded a highly significant difference (U = 179, p-value = 0.000). A p-value of zero represents the strongest possible evidence against the null hypothesis, indicating a clear distinction in scrape times between EE and 3P-IO.

• Precision (BLEU): The Mann-Whitney U test indicated a statistically significant difference in precision scores (U = 5997.5, p-value = 0.0135). This p-value suggests a less than 2% chance of this difference arising by chance, implying a meaningful difference in the precision between EE and 3P-IO.

• Load Success Rate: As load success data is binary (success or failure), it was transformed to a continuous scale (-1 for failure, 1 for success) for compatibility with the Mann-Whitney U test. This transformation allows the test to account for the magnitude of difference in success rates. The resulting U-value of 4700 and a statistically significant p-value of 0.0133 indicate a meaningful distinction in load success rates between the two programs.

Having established statistically significant differences between EE and 3P-IO using Mann-Whitney U tests, the next step is to determine which

program exhibits superior overall performance. TOPSIS is employed for this purpose, allowing for calculation of the relative closeness (RC) score to an ideal solution for each program. This RC score considers both desired and undesired parameter values, providing a comprehensive assessment of overall performance. In this study, the implementation of the TOPSIS method is using the Manhattan distance metric.

The TOPSIS analysis yielded a relative closeness (RC) score of 0.8101 for EE and 0.9245 for 3P-IO. A higher RC score indicates greater closeness to the ideal solution. Therefore, 3P-IO emerges as the better-performing program, suggesting it achieved a more favorable balance across the desired and undesired criteria compared to EE. While the TOPSIS analysis indicates 3P-IO's overall superiority, it is important to acknowledge its limitations. EE's 100% load success rate is a significant advantage, particularly for tasks requiring guaranteed content retrieval. A potential future direction could involve a hybrid approach: leveraging 3P-IO's speed and precision when load success is not critical and utilizing EE when content retrieval reliability is paramount.

Moreover, advanced tools like PuppeteerExtraSharp, especially with the Stealth Plug-in, demonstrate significant capabilities. In scenarios where the HTML document structure of a site is known in advance, extracting the main content through nodes becomes easily achievable. By adhering to ethical practices, such as incorporating delays between requests to avoid overloading servers, PuppeteerExtraSharp solutions maintain functionality without being blocked. This approach allows for efficient harvesting, achieving speeds of approximately 10,000 articles per 24 hours (utilizing Nereus, another program authored by the researcher).

Another efficiency-oriented approach involves preprogramming interactions, such as simulating user clicks on specific buttons, to achieve high-speed automation of real browser environments. This method also mimics human input on the PC, rather than relying on simulated instances of Chromium, thereby enhancing the automation process.

The efficacy of such analytical tools is ultimately measured by their relevance to real-world dynamics and impact on decision-making processes. To further underline the value of EE and 3P-IO, a sentiment chart is presented here tracking negative media narratives about Serbia from September 16 to October 3, 2023 (Fig. 4).

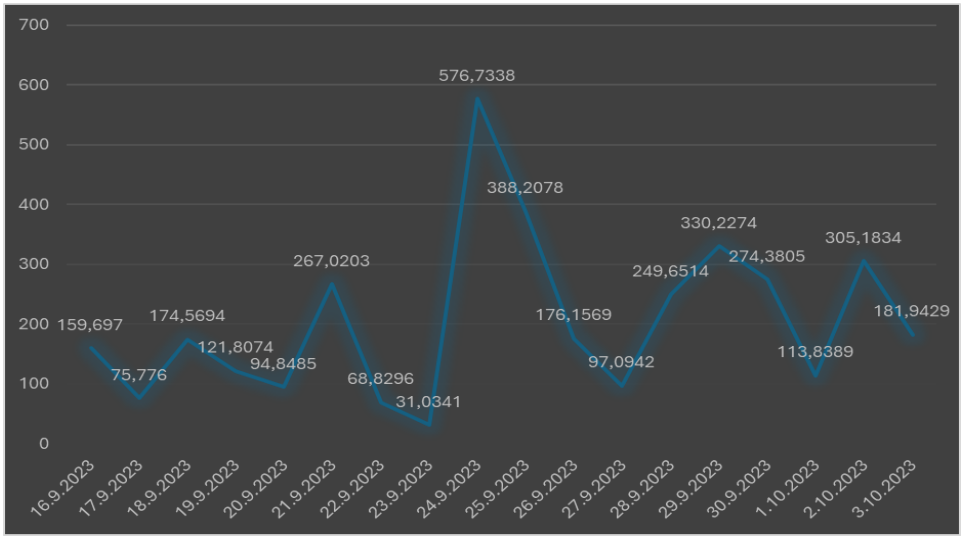
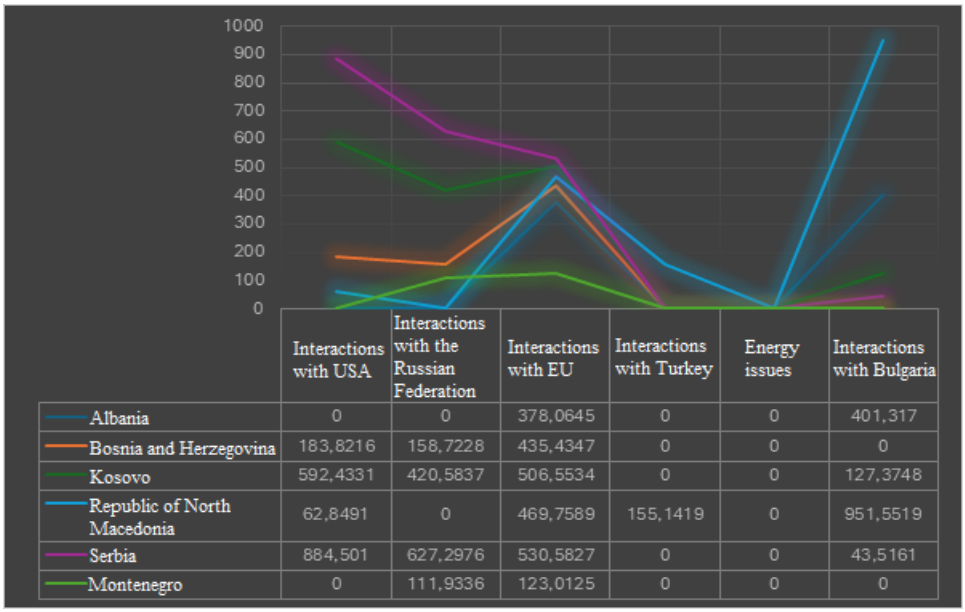


Figure 4. Negative Sentiment Trends: Serbian Media Landscape (Sept 16 – Oct 3, 2023)

The media analysis processed 3,459 articles, with 351 deemed relevant. The chart highlights a notable negative sentiment peak on 24.09.2023, directly correlating with the Banjska attack. This analytical tool provides a quantifiable method for measuring and visualizing crisis situations through media sentiment tracking. The implementation offers a compelling approach to evaluating regional security dynamics. The line chart (Fig. 5) depicts negative sentiment in content for each of the Western Balkans 6 (WB6) countries from September 27 to October 3, 2023 (the programmatically generated image underwent editorial correction for translation to English).



	Interactions with USA	Interactions with the Russian Federation	Interactions with EU	Interactions with Turkey	Energy issues	Interactions with Bulgaria
Albania	0	0	378,0645	0	0	401,317
Bosnia and Herzegovina	183,8216	158,7228	435,4347	0	0	0
Kosovo	592,4331	420,5837	506,5534	0	0	127,3748
Republic of North Macedonia	62,8491	0	469,7589	155,1419	0	951,5519
Serbia	884,501	627,2976	530,5827	0	0	43,5161
Montenegro	0	111,9336	123,0125	0	0	0

Figure 5. Comparative Negative Sentiment: Western Balkans Regional Media Analysis (Sept 27 – Oct 3, 2023)

Fig. 5 is generated based on 7,545 articles, of which 534 are considered relevant. The examples shown represent a fraction of the program's capabilities – creating visualizations across different topics along multiple binary classifications that go beyond sentiment analysis, time series, and anomaly detection. But this is sufficient to demonstrate that the question at the center of the study is not lacking importance.

CONCLUSION

This study investigated the performance of two web content extraction programs, EE and 3P-IO, through a comparative analysis. The evaluation employed a dataset of 100 webpages curated by EE to ensure balanced representation of regional security topics. Both programs were assessed on four key metrics: crawl time, scrape time, precision, and load success rate.

The findings revealed statistically significant differences between EE and 3P-IO using Mann-Whitney U tests, confirming the hypothesis outlined in the introduction. While further research using a wider range of data sources is recommended, the results suggest that 3P-IO exhibits a more favorable balance across the evaluated criteria. 3P-IO achieved faster crawl and scrape times with comparable precision to EE. However, EE's 100% load success rate offers a distinct advantage in scenarios where guaranteed content retrieval is paramount.

It is important to acknowledge that 3P-IO's fallback strategy can still achieve some level of extraction even with failed loads. Even so, a hybrid approach offers a promising avenue for future research. It could leverage 3P-IO's speed and precision advantages, while activating the computer vision component only when the fallback strategy fails to achieve satisfactory content retrieval. Research efforts aimed at validation of the capabilities of a hybrid approach in enhancing web content extraction for regional security applications, would be of value.

Overall, this study provides valuable insights into the strengths and limitations of two differentiated frameworks, empowering stakeholders to select the most appropriate tool for their specific needs. Additionally, the concept of a unified design lays the foundations for next-generation web crawlers that will inevitably be more integrated with large language models, able to "see" their environment and to autonomously decide upon the significance of gathered information while remaining entirely indistinguishable to human users. By exploring these avenues, we can continue to improve the efficiency and reliability of web content extraction, the evolution of the OSINT toolkit, ultimately supporting more robust and effective data-driven decision-making in regional security.

BIBLIOGRAPHY:

Атанасов, П. (2023). Закономерности при формирането на конспиративни теории. Слух, страх и активно комуникиране. *Сигурност и отбрана,* (1), 158-177. https://doi.org/10.70265/AMOP4068 // Atanasov, P. (2023). Zakonomernosti pri formiraneto na konspirativni teorii. Sluh, strah i aktivno komunikirane. *Sigurnost i otbrana,* (1), 158-177. https://doi.org/10.70265/AMOP4068

Василев, В. & Борисова, Р. (2023). Хибридно влияние на Русия и Китай в Сърбия. *Сборник доклади от научна конференция „Актуални проблеми на сигурността“*, 26-27.10.2023, НВУ „Васил Левски“, Велико Търново, 229-238. ISSN 2367-7473 // Vasilev, V. & Borisova, R. (2023). Hibridno vliyanie na Rusija i Kitaj v Srbija. *Sbornik dokladi ot nauchna konferentsiya „Aktualni problemi na sigurnostta“*, 26-27.10.2023, NVU „Vasil Levski“, Veliko Tarnovo, 229-238. ISSN 2367-7473

Andric, J., & Terzic, M. (2023). Intelligence Cycle in the Fight Against Terrorism with Usage of OSINT Data. *Journal of Information Systems & Operations Management, 17* (1), 1-16. https://web.rau.ro/websites/jisom/Vol.17%20No.1%20-%202023/JISOM%2017.1_1-16.pdf

Hayes, D., & Cappa, F. (2018). Open-source intelligence for risk assessment. *Business Horizons, 61* (5), 689-697. https://doi.org/10.1016/j.bushor.2018.02.001

Krotov, V., Johnson, L., & Silva, L. (2020). Tutorial: Legality and Ethics of Web Scraping. *Communications of the Association for Information Systems, 47*, 539-563. https://doi.org/10.17705/1CAIS.04724

Microsoft .NET. (n.d.). *ML.NET*. Retrieved November 19, 2024, from https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet

Ninov, M., & Nikolov, V. (2023). Ethno-demographic dimensions of national power. *Global Journal of Management and Business Research, 15* (4), 107-114. https://doi.org/10.59035/GJMB4806

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak & D. Lin (Eds.), *Proceedings of the 40th annual meeting on association for computational linguistics (ACL '02)* (pp. 311-318). Association for Computational Linguistics. https://aclanthology.org/P02-1040