Impact Factor 6.1

Journal of Cyber Security

ISSN:2096-1146

Scopus

DOI

Google Scholar

More Information
www.journalcybersecurity.com

Crossref doi Google Scholar scopus

Automated Scraping and Analysis of Social Media data for Cybercrime Investigation *

Dr Shubha Rao V
BMS College of Engineering
Bangalore, India

Bhuvan Savant
BMS College of Engineering
Bangalore, India

Arth Rawat
BMS College of Engineering
Bangalore, India

Aditya Kishore
BMS College of Engineering
Bangalore, India

Dr. Sneha Girish
KS Institute of Technology
Bangalore, India

I. Abstract

With the rapid development of the Internet, social networks have become one of the most widely used platforms. Users frequently share their moments, thoughts, and emotions through these channels. However, this surge in social media usage has also led to increased cybercrime in recent years. Cybercrime can manifest itself in various forms, including hate speech, bullying, scams, and more. Given the vastness of the Internet, manually collecting and analyzing such data is tedious, impractical, and time-consuming.

This research proposes a cross-platform tool designed to automate the scraping and analysis of a suspected cybercriminal's activity on multiple social media accounts.

It should be noted that all data used was collected with proper permission taken . The data used was from our close friends only. The data is neither stored nor shared with any third party

II. Background

Cybercriminals increasingly exploit social media platforms like Twitter, Facebook, and Instagram to obtain information about the sites. spreading rumors and misinformation, promoting illegal activities, or coordinating cybercrimes directly or indirectly.

1. Challenges

Due to legal restrictions and security rules imposed by social media platforms, it is difficult to directly scrape data from such sites. Hence collecting data was the first and most challenging step of our research. Different tech- niques have been introduced to ease this process like [13] involves video and audio analysis to extract features like facial expressions, body language, and speech patterns.

2. Platforms covered

In the current version of our tool, more focus has been on platforms that offer relatively accessible data extraction pathways. Other platforms, such as WhatsApp, employ end-to-end encryption, making data retrieval nearly impossible without device-level access. Similarly, platforms like Instagram enforce strict rate-limiting policies and automated IP bans, which significantly hinder scrap- ing attempts. These limitations necessitated a selective approach in choosing platforms for inclusion in our system.

- X (formerly known as Twitter)
- Telegram
- Reddit
- News scraper

3. Aim

The objective is to develop a user-friendly, cross-platform tool capable of collecting and analyzing data from the aforementioned social media platforms. In ad- dition to streamlining the data acquisition and analysis process, the aim is to minimize resource consumption for each operational stage. Furthermore,secure authentica- tion mechanisms need to be incorporated to ensure con- trolled and authorized access during data collection.

III. Related Work

Related research papers in this field have been surveyed thoroughly before diving into any implementation. Many scraping methods have been suggested in recent times with various datasets. Most of the papers found were mainly based on the analysis of the collected data.

For instance, [2] focused on using contextual embeddings generated by BERT to classify the data, whereas [4] used TF-IDF embeddings for the same. Research has also been carried out focusing on data from a particular region; [8] extracted tweets of Nigerian origin and processed them for sentiment analysis.

As social media data is a mix of different media, capturing every aspect of content is essential for forensics. The authors of [13] suggested a machine learning-based method combining computer vision and natural language processing techniques to analyze and classify personification patterns. Similarly, [14] addressed the limitations of existing sentiment models by introducing a dual-channel network combining BiLSTM and GCN with dependency parsing for better syntactic understanding of informal social media texts.

Furthermore, [16] directly aligns with our objective, as it addresses the challenge of collecting data efficiently from social networks for criminal investigations. Their approach, which combines automated web browsers with third-party APIs, inspired our data collection mechanism. It demonstrates that hybrid crawling methods can achieve timely and rich data extraction, fetching up to 9,802 account elements per user in about 12.79 minutes.

Additionally, papers like [4], [5], and [6] focus on cy- ber security contexts. These works show the potential of NLP in processing large-scale textual data for threat de- tection and emphasize the need for automated solutions to manage streaming content. Particularly, [5] and [6] high- light the benefits of combining neural architectures like CNNs and RNNs with NLP models for accurate anomaly detection and classification.

While these papers delve into different aspects of social media analytics, a major gap still exists in the domain of robust, adaptive, and secure data collection mechanisms. Most prior work assumes the availability of data and focuses only on downstream analysis. Our work aims to bridge that gap by developing a cross-platform tool that emphasizes scalable and secure data extraction as a foundational step.

The next section explores the challenges faced with extractign data from each platform and the corresponding techniques which were devised.

IV. Methodology

As mentioned earlier, social media platforms impose several countermeasures to prevent automated data scraping. These include rate limiting, IP address blocking, frequent changes in HTML structures and increasingly strict authentication mechanisms. Consequently, collecting data from such platforms is often either highly restricted or requires workarounds that are short-lived. Many available methods either stop working after a short period due to updates on the platform or require manual intervention to maintain. Recognizing these limitations, our objective was to identify and develop sustainable scraping techniques that would be more robust to platform-level changes and viable for long-term use in research and investigative applications.

1. X(Formerly Twitter)

X (formerly Twitter) proved to be one of the most challenging platforms during the implementation phase, primarily due to its aggressive countermeasures against automation and its frequent structural and policy updates (see Fig. I). Initial efforts involved exploring the use of the official X API for data extraction; however, after evaluating the cost-to-service ratio, it was determined that the available pricing tiers did not align with the requirements of the research, particularly given the limited data access and rate restrictions imposed under the basic plans. Consequently, focus shifted toward developing a more resilient solution capable of ensuring reliable and scalable data collection.

Early strategies employed widely used libraries such as BeautifulSoup and Selenium for automating data scraping. However, these tools were found to be highly vulnerable to anti-bot measures, notably IP rate-limiting and account bans. While proxy servers with frequent IP rotation were considered to mitigate this issue, a more sustainable and low-maintenance solution was ultimately preferred. It was observed that X employs XHR (XMLHttpRequest) mechanisms to dynamically load data without necessitat- ing a full page reload, thus modifying the HTML content on the fly. Leveraging this insight, the collection process was adapted to intercept and store XHR requests, allow- ing the extraction of comprehensive information from a single tweet into a structured JSON format using a headless browser — a browser operating without a graphical user interface (as depicted in Fig. I).

The initial design aimed to extend this approach by implementing a dynamic scrolling mechanism to simulate user behavior and gather data from multiple tweets. However, this introduced additional complexities, including the need for timeout controls to ensure complete loading of tweet content prior to extraction. Despite these measures, inconsistencies and unexpected failures persisted. As a contingency, an alternative approach involving the capture of tweet screenshots and compilation into a PDF format was explored. Nevertheless, this method was found to impede downstream AI processing, as models encountered difficulties in interpreting mixed textual and visual content, often resulting in hallucinations or misclassifications.

To address these challenges, the scraping pipeline was restructured using a multi-threaded architecture (illustrated in Fig. I). In the final approach, the system first scrolled through a user's timeline to collect individ- ual tweet URLs, followed by the initiation of multiple threads, each responsible for parsing a single tweet us-

ing headless browser logic. This multi-threaded execution significantly enhanced the overall efficiency and reliabil- ity of data extraction, enabling the collection of tweet content—including textual and visual components—into a standardized, AI-readable JSON format.

Despite these improvements, one limitation remained: the system was unable to support data extraction from unverified users. This constraint arose due to enforced login or signup prompts on X, which disrupted automation and increased the risk of account flagging or access blocking.

Then we used a local model of Ollama which was ollama3.2 with 1 billion parameters . This can be replaced by any model/api according to budget and number of requests. We can also further fine-tune the model for more detailed detection.

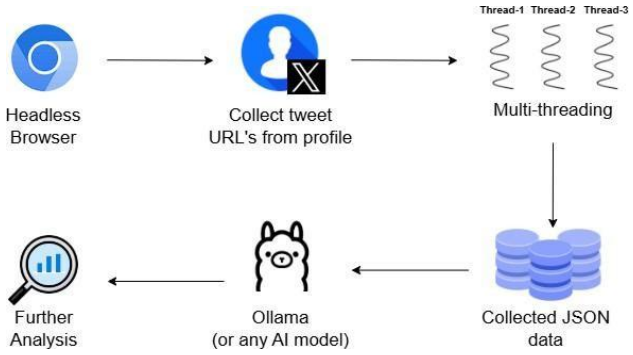

FIGURE I: WORKFLOW FOR X DATA COLLECTION
The system uses a headless browser and multi-threading to collect and process tweet data efficiently.

article amsmath inputenc [T1]fontenc enumitem

2. Integration with Telegram

The integration process for Telegram was reported to involve a distinct set of challenges compared to web-centric platforms, primarily because Telegram relies on a dedicated client API rather than a readily scrape-able web interface for extensive message data access. The approach adopted consequently focused on the robust utilization of this official API, integrated within the project's desktop application framework.

[label=0., wide, labelindent=0pt]**API Integration and Authentication:** The core strategy involved leveraging the official Telegram Client API. After evaluating available Python libraries, Telethon was selected based on its comprehensive feature set, asynchronous capabilities essential for non-blocking network I/O within the GUI, and active maintenance status. It was noted that Telegram's API usage model, requiring developer credentials (API ID and Hash) and user authentication (phone number, login code/password), suited applications intended for users accessing their *own* data, unlike some platforms where API access might be restrictive. A key challenge identified during this phase was managing the authentication flow, which Telethon handles interactively, and securely storing the user's session state locally (via a session name file). **Asynchronous Operations within a Synchronous GUI:** A significant technical hurdle reported was the integration of Telethon's asynchronous nature with the synchronous event loop of the PyQt5 desktop GUI. Direct calls to asynchronous Telethon functions from standard PyQt5 signal handlers would block the user interface. To mitigate this, a bridging mechanism was implemented using PyQt5's QTimer. This timer periodically yielded control to the asyncio event loop, enabling background network operations (like message fetching or media downloading) while maintaining GUI responsiveness, thereby ensuring a fluid user experience. **Data Extraction and Structur- ing:** Data extraction was accomplished programmatically through specific Telethon functions within an asynchronous workflow. The process involved:

123. • client.start() for handling authentication.

- client.get dialogs() for retrieving the user's recent chats.
- client.get messages() for fetching messages within chats, iterated up to a defined limit (initially set low for performance considerations).
- msg.download_media() for downloading associated media directly to a local directory.

It was observed that, unlike web scraping scenarios requiring HTML parsing or network request monitoring, Telethon provided direct access to structured message objects. These objects were processed immediately upon retrieval to extract relevant fields (such as chat name, sender ID, message text, and media information). This extracted data was aggregated into an in-memory list of Python dictionaries (self.scraped data), creating a structured format suitable for display, processing, or export.

4. **Integrated Content Analysis (Abuse Detec- tion):** Simultaneously with data extraction, a sim- ple keyword-based AbusiveLanguageDetector was incorporated into the pipeline. The text of each retrieved message was passed to this detector. The outcome, including a boolean flag indicating detection and details of matched terms/positions, was stored alongside the message data within the self.scraped data structure. This integration facilitated the immediate flagging of content according to a user-defined keyword list, serving the goal of providing a basic, user-controllable content filter-

ing mechanism directly within the data acquisition process.

5. **Export and Reporting:** Recognizing the need for diverse downstream applications, multiple export options were implemented:
    - **CSV Export:** This option converted the structured self.scraped data list directly into a tabular CSV format, suitable for spreadsheet analysis or batch processing tasks.
    - **PDF Report:** A human-readable report was generated using the WeasyPrint library. This involved dynamically creating an HTML document populated with the messages, embedding downloaded images, and employing CSS to visually highlight terms identified by the abuse detector. A section containing summary statis- tics was also included. While noted as less ideal for direct AI ingestion compared to raw text or CSV, this format was deemed valuable for contextual review.
    - **Screenshot:** A basic utility was included to capture the application's visual state.
6. **Limitations and Considerations:** Despite the stability offered by using the official API via Telethon, several limitations were acknowledged:
    - **Authentication Requirement:** The tool necessitates active user login and API credentials, precluding anonymous use.
    - **Scope Limitation:** The implementation primarily focused on recent messages due to the use of limit parameters. Accessing complete chat histories would necessitate the implementation of pagination logic to comply with Telegram's API limits for bulk data retrieval.
    - **Detection Simplicity:** A core limitation remained the naive, keyword-based nature of the abuse detection, which lacks contextual understanding (as further discussed in Section 6: Discussion).
    - **Dependency:** The tool's functionality relies on the continued availability and consistent policies of both the Telegram API and the Telethon library.
7. **Downstream Processing Potential:** The structured data output, particularly the message text available via CSV or direct processing, was identified as suitable input for more sophisticated Artificial Intelligence models. It was suggested that this extracted text could be fed into local models via frameworks like Ollama (e.g., using models such as llama3, mistral, or specialized fine-tuned versions) or utilized with cloud-based Natural Language Processing APIs. Such downstream processing could enable advanced tasks like nuanced sentiment analysis, topic modeling, or more reliable toxicity detection, thereby building upon the initial keyword-based flagging provided by the developed tool.

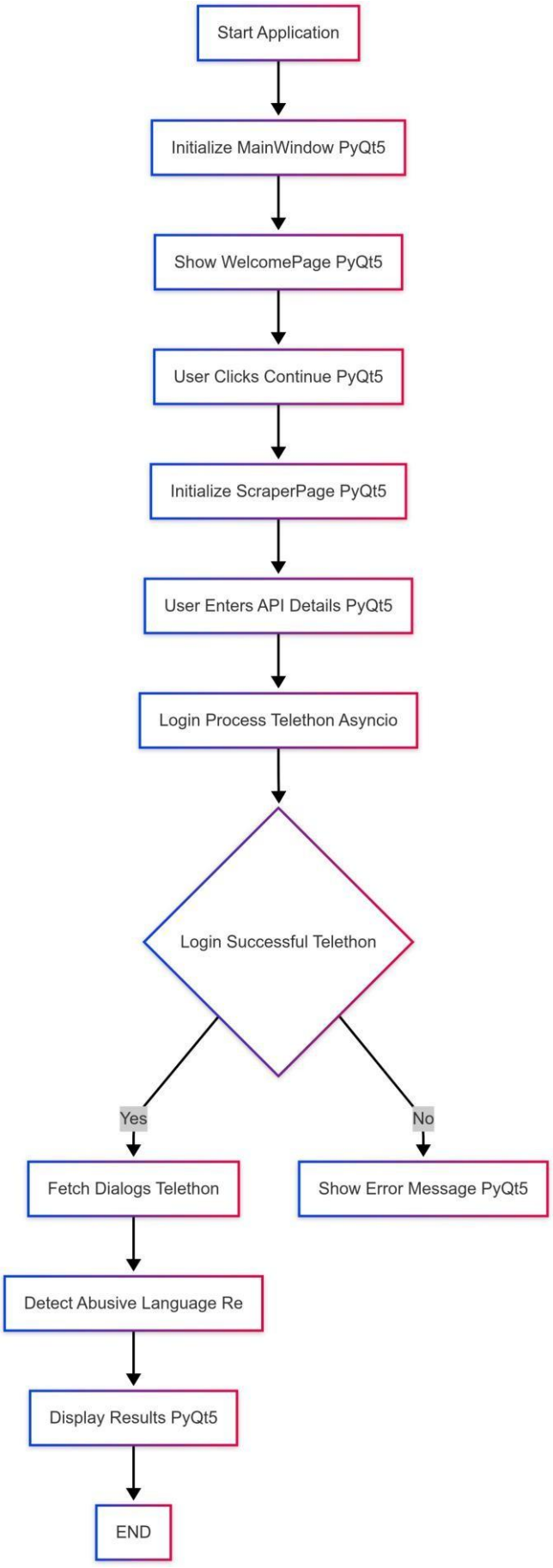

FIGURE II: WORKFLOW FOR TELEGRAM SCRAPPING

3. Reddit(forum social media platform)

Reddit, a widely used social media platform, was described as presenting unique challenges and opportunities during the implementation phase. Unlike some other platforms, Reddit was noted for offering a relatively open API, which facilitated data extraction efforts. However, the platform's vast and diverse content required careful consideration in terms of data handling and analysis.

Initially, the team leveraged the official Reddit API through the PRAW (Python Reddit API Wrapper) li- brary. This approach allowed access to a wide range of data, including user posts, comments, and subreddit information. The API's read-only mode was particularly beneficial, as it enabled the gathering of data without the risk of violating Reddit's terms of service or triggering anti-bot measures.

One of the primary challenges faced during the project was the sheer volume of data available on Reddit. With millions of active users and countless posts and comments, it was noted as crucial to implement efficient data filtering and analysis techniques. The team focused on extracting posts and comments from specific users, which required careful handling of API rate limits and pagination.

To enhance data analysis capabilities, a keyword and potentially malicious content detection system was implemented. This involved scanning posts and comments for predefined keywords and phrases that could indicate harmful or inappropriate content. The system was designed to be flexible, allowing for easy updates to the list of keywords and phrases as needed.

Despite the advantages of using the Reddit API, cer- tain limitations were encountered in terms of data completeness and real-time access. The API's rate limits and restrictions on certain types of data meant that specific data points had to be prioritized and strategies had to be implemented to manage data collection efficiently.

To address these challenges, alternative data collection methods, such as web scraping, were explored. However, given Reddit's dynamic content and frequent updates, this approach proved to be less reliable and more prone to errors. As a result, the focus shifted towards optimizing the use of the API and developing robust data processing pipelines to handle the accessible data.

In terms of AI integration, a local model, Ollama 3.2, with 1 billion parameters, was utilized to analyze the extracted data. This model was chosen for its balance between performance and resource requirements. Depending on budget and data volume, alternative models or APIs could be employed, and further fine-tuning could enhance detection accuracy.

Overall, the experience with Reddit highlighted the importance of leveraging official APIs where possible, while also emphasizing the need to adapt to platform-specific

challenges. By combining efficient data extraction techniques with advanced AI models, valuable insights were gained from Reddit's vast user-generated content.

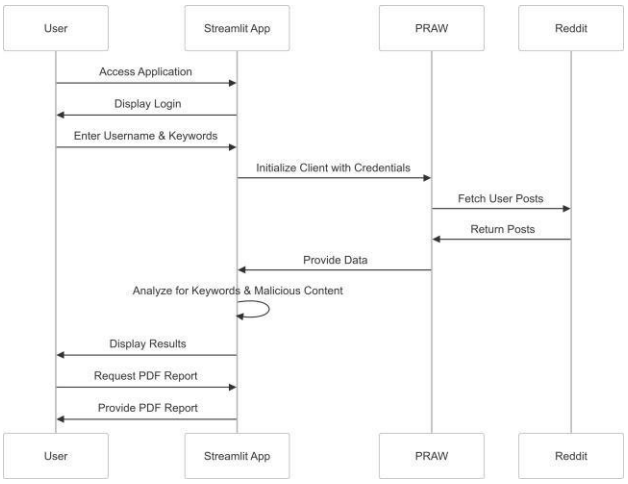

FIGURE III: WORKFLOW FOR REDDIT SCRAPPING

The sequence diagram illustrates the interaction between the user, Streamlit application, PRAW library, and Reddit during the data extraction and analysis process. It highlights steps from user input to data fetching, analysis for malicious content, and generation of a PDF report.

4. News Scraper

An Advanced News Scraper was developed as a sophisticated tool intended to fetch, analyze, and report on news articles from diverse sources. The system was reported to leverage multiple technologies and libraries to provide comprehensive insights into news content, including the detection of potentially abusive language and AI-driven analysis. The key components and functionalities implemented are detailed below.

System Architecture The system architecture was based on Streamlit, a framework selected for creating interactive web applications in Python. Integration with several libraries and APIs was undertaken to perform the required tasks:

- **Streamlit**: Employed for building the user interface and managing user interactions.
- **BeautifulSoup and Requests**: Utilized for web scraping activities, specifically fetching HTML content from target web pages.
- **Feedparser**: Incorporated to parse RSS feeds obtained from various news sources.
- **ReportLab**: Used for the automated generation of PDF reports summarizing the news analysis findings.
- **Google Generative AI (Gemini)**: Integrated to provide AI-driven analysis of news content, facilitat- ing the detection of potentially abusive language and the generation of contextual insights.
- **Scikit-learn**: Applied for text processing tasks, including similarity analysis aimed at identifying potentially duplicate news items.

Functionality Several key functionalities were implemented within the Advanced News Scraper:

- **News Fetching and Parsing:** The system was designed to fetch news articles from predefined RSS feeds, encompassing sources such as CNN, BBC, and Reuters, alongside regional sources in Hindi and Kannada. Fetched articles were subsequently parsed to extract essential metadata including titles, descriptions, publication dates, and source links.
- **Abusive Content Detection:** A predefined set of abusive keywords across multiple languages (English, Hindi, Kannada) was defined within the sys- tem to enable the detection of potentially harmful content. These keywords were highlighted visually within the displayed news articles. Furthermore, the system was capable of utilizing the integrated Gemini AI model to analyze text for abusive content beyond the limitations of predefined keywords, offering a more nuanced detection capability.
- **AI-Driven Analysis:** Integration with Google's Gemini AI enabled the system to perform advanced analysis of the news content. This analysis included identifying main themes, emerging trends, potential biases, and assessing regional or global impacts conveyed in the articles. The AI model was configured using an API key, and its generated responses were parsed programmatically to extract relevant analyt- ical insights.
- **PDF Report Generation:** The capability to generate comprehensive PDF reports summarizing the collected news articles and the corresponding AI analysis was included. These reports incorporated highlighted abusive content, publication details, and AI-generated insights, formatted using the ReportLab library.
- **Duplicate Detection and Statistics:** Text simi- larity analysis, based on TF-IDF vectors and cosine similarity from Scikit-learn, was employed to iden- tify potential duplicate news items primarily based on title similarity. Additionally, summary statistics regarding news coverage, such as source distribution and the measured prevalence of detected abusive content, were provided to the user.

User Interaction The user interface, built with Streamlit, was designed to allow users to configure the news fetching process by selecting date ranges and lan- guages of interest, and optionally specifying keywords for filtering. Users were provided with options to enable or

disable the AI analysis component for abusive content detection. The interface included controls for initiating news fetching, triggering analysis with the Gemini model, generating PDF reports, and displaying news coverage statistics.

5. Challenges and Considerations

Several challenges were noted during the implementation and deployment of the Advanced News Scraper:

- **API Configuration:** Proper configuration and validation of the Gemini API key were identified as cru- cial for the AI analysis functionality. The system incorporated checks for API key validity and network connectivity to the service.
- **Language Support:** Handling multiple languages necessitated careful management of character encoding standards and the definition of language-specific keyword sets for abuse detection.
- **Performance:** Fetching and analyzing potentially large volumes of news articles was recognized as po- tentially resource-intensive. Efforts were made to use efficient data structures and processing algo- rithms to manage performance.
- **Privacy and Data Usage:** The system was de- signed to process news content primarily in memory and was not intended to store persistent user data beyond session state, aligning with common privacy standards.

Overall, the Advanced News Scraper served as a demonstration of integrating web technologies, natural language processing techniques, and generative AI to create a comprehensive tool for news analysis. Its modular design was intended to allow for straightforward extension and adaptation to accommodate new requirements or data sources in future iterations.

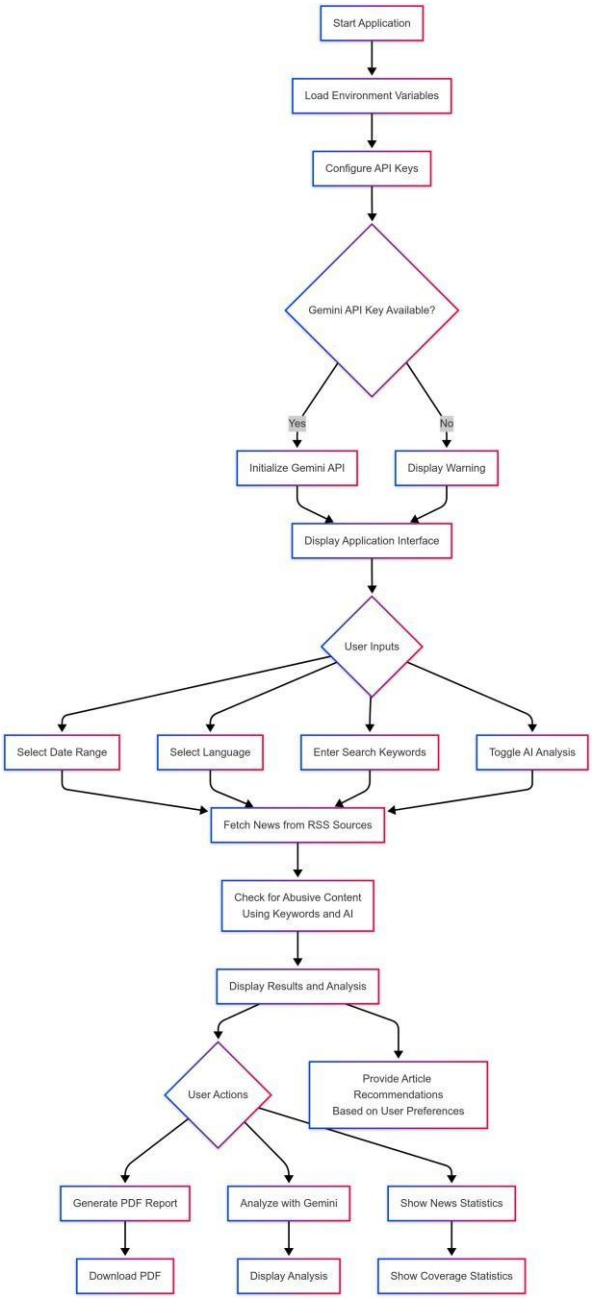

FIGURE IV: WORKFLOW FOR NEWS SCRAPING

The flowchart illustrates the workflow of a news analysis application, detailing processes from application startup and API configuration to user input handling, news fetching, content analysis, and output generation, including PDF reports and AI-driven insights. It highlights decision points and user interaction paths throughout the system.

V. CONCLUSION

The scraping of social media platforms was successfully automated, and the collected data was integrated into AI models for detailed analysis. The developed approach demonstrated the feasibility of constructing an end-to-end system capable of efficient data extraction and intelligent content interpreta- tion. The results suggested that the combination of automated data collection with advanced AI techniques can significantly

enhance the depth and accuracy of social media analytics.

Future research could focus on fine-tuning the AI models to improve the granularity and accuracy of sentiment analysis across diverse linguistic styles and regional variations. Incorporating layout detection models was identified as a promising direction for selectively focusing on specific regions of a tweet or post, thereby increasing information density and relevance during analysis. Additionally, the exploration of multi-modal fusion techniques—integrating textual, visual, and metadata components—could further strengthen the contextual understanding of social media content, making the system more effective for investigative and analytical applications.

Acknowledgement

We thank Dr Shubha Rao V for her support in this research
.

Authors' Contributions

All authors have participated in drafting the manuscript. All authors read and approved the final version of the manuscript.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability

The data supporting the findings of this study are neither stored anywhere and should be only used after proper permission has been granted.

Ethical Statement

In this article, the principles of scientific research and publication ethics were followed. This study did not involve human or animal subjects and did not require additional ethics committee approval.

Declaration Of AI Usage

No AI tools were used in the creation of this manuscript.

References

[1] Li, Ying, et al. "NEDetector: Automatically extracting cybersecurity neologisms from hacker forums." Journal of Information Security and Applications 58 (2021): 102784.

[2] Sandescu, Cristian, et al. "Extracting exploits and attack vectors from cybersecurity news using NLP." UPB Sci. Bull., Series C 84.2 (2022).

[3] Vadapalli, Satyanarayan Raju, George Hsieh, and Kevin S. Nauer. "Twitterosint: automated cybersecurity threat intelligence collection and analysis using twitter data." Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2018.

[4] Mohammed, Sahar Yousif, and Mohammad Aljanabi. "From Text to Threat Detection: The Power of NLP in Cybersecurity." SHIFRA 2024 (2024): 1-7.

[5] Lughbi, Huda, Mourad Mars, and Khaled Almotairi. "A Novel NLP-Driven Dashboard for Interactive CyberAttacks Tweet Classification and Visualization." Information 15.3 (2024): 137.

[6] Malik, Saima, and Nadir Ali. "Automating Cyber-Security Prevalence Response Using Intelligent Retrieval and Neural Networks."

[7] Jünger, Jakob. "Scraping social media data as platform research: A data hermeneutical perspective." 86272 12 (2023): 427-441.

[8] Kolajo, Taiwo, et al. "A framework for pre-processing of social media feeds based on integrated local knowledge base." Information processing & management 57.6 (2020): 102348.

[9] KASHKIN, Vasily, and Yuriy ANDROSIK. "USING SOCIAL MEDIA ANALYTICS AND PARSING PROGRAMS TO IDENTIFY AND ANALYzE THE TARGET AUDIENCE." Business Excellence & Management 14.1 (2024).

[10] Sowmya, C. M., S. Deena, and S. Anbuchelian. "Performance of sentimental analysis by studying and mining social media using parsing technique." 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT). IEEE, 2018.

[11] Feng, KJ Kevin, et al. "Mapping the Design Space of Teachable Social Media Feed Experiences." Proceedings of the CHI Conference on Human Factors in Computing Systems. 2024.

[12] Benson, Edward, Aria Haghighi, and Regina Barzilay. "Event discovery in social media feeds." Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies. 2011.

[13] Gawade, Prema Pandurang, and Sarang Achyut Joshi. "Algorithm for safety decisions in social media feeds using personification patterns." J. Adv. Inf. Technol 14.1 (2023): 145-152.

[14] Yin, zeyu, et al. "DPG-LSTM: an enhanced LSTM framework for sentiment analysis in social media text based on dependency parsing and GCN." Applied Sciences 13.1 (2022): 354.

[15] Singkul, Sattaya, et al. "Parsing thai social data: A new challenge for thai nlp." 2019 14th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP). IEEE, 2019.

[16] Kolajo, Taiwo, et al. "A framework for pre-processing of social media feeds based on integrated local knowledge base." Information processing & management 57.6 (2020): 102348.