International Journal of Computing and Artificial Intelligence

E-ISSN: 2707-658X
P-ISSN: 2707-6571
www.computersciencejournals.com/ijcai
IJCAI 2024; 5(1): 102-108
Received: 16-01-2024
Accepted: 19-02-2024

Mayank Kumar Tiwari
NIMS Institute of Computer Science & Technology, NIMS University Rajasthan, Jaipur, Rajasthan, India

Rahul Pal
Department of Pharmaceutics, NIMS Institute of Pharmacy, NIMS University Rajasthan, Jaipur, Rajasthan, India

Vansh Chauhan
NIMS Institute of Computer Science & Technology, NIMS University Rajasthan, Jaipur, Rajasthan, India

Vinit Singh
NIMS Institute of Computer Science & Technology, NIMS University Rajasthan, Jaipur, Rajasthan, India

Vijendra Singh
NIMS Institute of Computer Science & Technology, NIMS University Rajasthan, Jaipur, Rajasthan, India

Dr. Sasikala Dhamodaran
NIMS Institute of Computer Science & Technology, NIMS University Rajasthan, Jaipur, Rajasthan, India

Dr. Swati Sharma
Director, NIMS Institute of Engineering & Technology, NIMS University Rajasthan, Jaipur, Rajasthan, India

Corresponding Author:
Mayank Kumar Tiwari
NIMS Institute of Computer Science & Technology, NIMS University Rajasthan, Jaipur, Rajasthan, India

A python programming widely utilized in the development of a twitter bot as a sophisticated advance technical tool

Mayank Kumar Tiwari, Rahul Pal, Vansh Chauhan, Vinit Singh, Vijendra Singh, Dr. Sasikala Dhamodaran and Dr. Swati Sharma

DOI: https://doi.org/10.33545/27076571.2024.v5.i1b.88

Abstract
This abstract explores the utilization of Python programming as a pivotal tool in the development of advanced Twitter bots. With the rise of social media automation, Twitter bots have become indispensable for tasks ranging from content curation to real-time data analysis and engagement. Python, known for its simplicity, versatility, and robust libraries, emerges as the preferred programming language for crafting sophisticated Twitter bots. The paper delves into the technical intricacies involved in creating Twitter bots using Python, focusing on key aspects such as data scraping, natural language processing (NLP), sentiment analysis, and machine learning integration. These functionalities enable bots to gather relevant information from tweets, analyze user sentiments, and respond intelligently based on predefined criteria or learning algorithms. The bot's functionalities are authenticated using Twitter API keys and access tokens, with robust error handling mechanisms implemented for smooth operation. Through careful analysis of engagement metrics and audience insights, the bot aims to enhance user engagement, drive content curation, and provide a personalized experience for followers. This research article focus on the basic introduction for python programming involved in the twitter bot development, their scope and importance, pros and cons, methodology/implementation and results & analysis in the brief with the code of involving as python programming.

Keywords: Python, language, bot, web-technology, computer, programming language, technology, information, twitter-bot, coding

Introduction
The introduction sets the stage for the Python Twitter bot project, providing an overview of its purpose, goals, and significance. It highlights the increasing importance of automation in social media management and introduces the Tweepy library as a powerful tool for building Twitter bots. The introduction also outlines the key functionalities the bot will perform, such as posting scheduled tweets, responding to mentions, and retweeting relevant content. Additionally, it briefly mentions the project's structure, including sections on authentication, bot functionality, error handling, deployment, and maintenance [1]. The flowchart form the twitter bot as per the Fig. 1 as below section.
Creating a Twitter bot involves designing a program that autonomously generates and posts content on Twitter [1-2]. The bot can perform various tasks, such as sharing updates, interacting with users, or curating content based on specific criteria. It requires programming skills, knowledge of Twitter API usage, and consideration of ethical guidelines to ensure responsible automation and user engagement [3]. Overall, the introduction primes the reader for what to expect in the subsequent sections of the documentation.

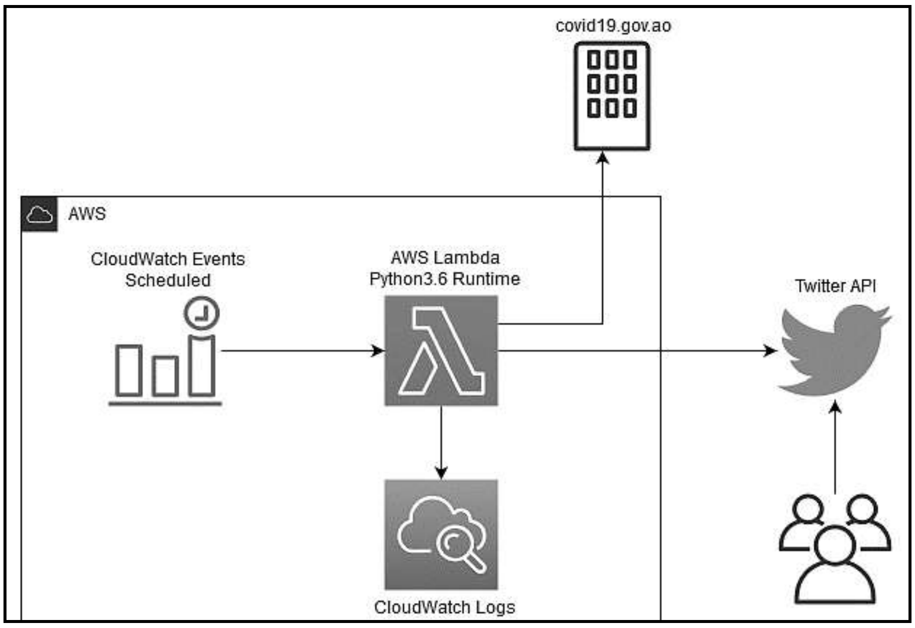


Fig 1: The flow chart for the python twitter bot creation

Purpose & Goals: The purpose of a Python Twitter bot is to automate certain tasks on the Twitter platform, leveraging the capabilities of the Twitter API and Python programming language. The primary goals typically include increasing engagement with followers by posting regular updates, sharing relevant content based on predefined criteria, and responding to user interactions in a timely manner. Python Twitter bots are designed to automate tasks on Twitter, saving time and effort for users [4-5]. Their primary goals involve:

- **Content Management:** Scheduling tweets, retweeting relevant content, and promoting specific hashtags or campaigns.
- **Engagement:** Responding to mentions and messages, providing 24/7 customer service, and fostering interaction with followers.
- **Data Gathering:** Analyzing Twitter data to gain insights into trends, user behavior, and public opinion. By leveraging Python's user-friendly syntax and libraries like Tweepy, these bots can efficiently handle repetitive tasks, increase a Twitter account's reach, and provide valuable data for further analysis.

Additionally, the bot may be designed to gather and analyze data, such as user feedback or trending topics, to inform decision-making and improve content strategy [6]. Ultimately, the Python Twitter bot aims to enhance the overall user experience, increase brand visibility, and streamline social media management efforts. The Table 1 outlining the steps involved in creating a Python Twitter bot along with their descriptions as follows.

Table 1: The list of steps involving in the creating Python Twitter bot with their description

Step	Description
Define Goals	Determine the bot's purpose, functionalities, target audience, and desired outcomes to guide development.
Set Up Twitter API	Register a Twitter Developer account, create an application, and obtain API keys and access tokens for authentication.
Install Libraries	Install Python libraries such as Tweepy for Twitter API interaction, and any additional libraries for bot functionality.
Authenticate	Use API keys and access tokens to authenticate the bot with Twitter, allowing it to access and interact with the API.
Develop Features	Write Python code to implement bot functionalities like posting tweets, replying to mentions, retweeting, etc.
Error Handling	Implement robust error handling to manage API rate limits, network errors, and other unexpected issues gracefully.
Test & Debug	Test the bot's functionalities thoroughly, debug any issues, and ensure proper functionality and performance.
Deployment	Choose a hosting platform like Heroku or AWS, configure deployment settings, and deploy the bot to make it operational.
Monitoring	Set up monitoring tools to track bot performance, engagement metrics, and handle any maintenance or updates as needed.
Documentation	Document the bot's functionalities, codebase, deployment process, and maintenance procedures for future reference.

These steps provide a structured approach to developing and deploying a Python Twitter bot, ensuring its functionality, reliability, and scalability. The bot can be designed to operate based on predefined rules, criteria, or triggers, allowing it to perform actions autonomously without manual intervention [6-7]. Python's simplicity and versatility make it a popular choice for developing Twitter bots due to its ease of use, extensive libraries for working with APIs, and robust community support.

Scope of The Work and Its Importance: The scope of work for a Python Twitter bot encompasses a wide range of functionalities and capabilities, depending on the specific goals and requirements of the project. This includes tasks such as automating posting schedules, curating and sharing content based on keywords or user interactions, monitoring mentions and hashtags, analyzing engagement metrics, and even implementing machine learning algorithms for sentiment analysis or personalized recommendations [8]. The scope of the Python Twitter bot project encompasses several key areas Fig. 2:

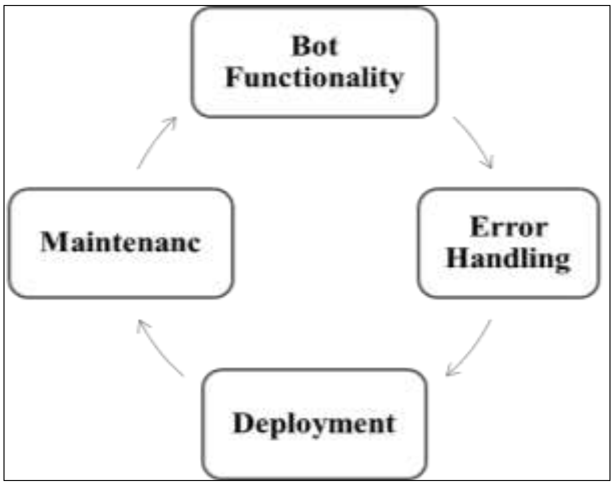


Fig 2: The scope of Python Twitter bot project

Bot Functionality: Designing and implementing features such as scheduled tweeting, replying to mentions, retweeting, and direct message responses involves creating algorithms and workflows that enable the bot to perform these actions autonomously. For example, scheduled tweeting may require a scheduler module that queues tweets based on specified times, while replying to mentions involves monitoring the Twitter feed for mentions and crafting appropriate responses based on predefined rules or keywords. Retweeting and direct message responses similarly require logic to identify relevant content and respond accordingly [9].

Error Handling: Developing robust error handling mechanisms to ensure the bot operates smoothly and gracefully handles unexpected situations, such as rate limits or network errors. Developing robust error handling mechanisms ensures that the bot operates smoothly even in the face of unexpected situations. This includes handling rate limits imposed by the Twitter API to prevent excessive requests, dealing with network errors that may interrupt communication with the Twitter servers, and gracefully handling any other unforeseen errors that may arise during execution. Proper error handling prevents the bot from crashing or malfunctioning and maintains its reliability and stability.

Deployment: Exploring deployment options and providing guidance on hosting the bot on platforms like Heroku or AWS, ensuring it remains accessible and operational. Exploring deployment options involves determining the best hosting platform for the bot, such as Heroku, AWS, or other cloud services. This includes configuring the environment, setting up necessary dependencies, and ensuring the bot's codebase is compatible with the chosen hosting solution. Guidance on deployment also includes considerations for scalability, security, and accessibility to ensure the bot remains operational and accessible to users.

Maintenance: Documenting best practices for monitoring bot performance, handling updates, and addressing maintenance tasks to sustain its functionality over time. Documenting best practices for maintaining the bot involves outlining strategies for monitoring its performance, handling updates to the codebase or API dependencies, and addressing any maintenance tasks that may arise. This includes setting up monitoring tools to track key metrics like response times, error rates, and API usage limits, as well as establishing procedures for testing and deploying updates to the bot's functionality. Regular maintenance ensures the bot remains functional, up-to-date, and capable of meeting its intended objectives over time [10-11].
The importance of a Python Twitter bot lies in its ability to streamline social media management, save time and effort by automating repetitive tasks, increase user engagement through timely and relevant interactions, gather valuable data for analytics and decision-making, and enhance brand visibility and reputation through consistent and targeted communication strategies. A well-designed Python Twitter bot can significantly contribute to an organization's social media marketing efforts and overall digital presence. The importance of this work lies in its ability to streamline social media management for individuals and businesses [12]. Additionally, the project demonstrates the practical application of Python programming skills and showcases the versatility of the Tweepy library in building custom Twitter solutions. The Python Twitter bot project serves as a practical tool for enhancing user engagement and maximizing the impact of online presence.

Pros & cons

Python Twitter bots offer several advantages, including automation of tasks like posting tweets and responding to mentions, flexibility in design and customization, strong community support, scalability for increased workload, and integration capabilities with other platforms. They also come with challenges such as managing API limitations to avoid suspension, adhering to ethical guidelines, regular maintenance requirements for optimal performance, complexity in developing advanced features, and the need for robust security measures to protect against potential vulnerabilities and misuse [9-12]. The several key points as pros for the python twitter bot discussed as below followings:

Automation: Saves time and effort by automating repetitive tasks such as posting tweets, responding to mentions, and retweeting relevant content. Python Twitter bots automate tasks like posting tweets, responding to mentions, and retweeting, reducing manual effort and allowing users to focus on more strategic activities.

Enhanced Engagement (EE): Increases user engagement by promptly responding to mentions and direct messages, thereby fostering interaction and building a stronger online presence. These bots improve user engagement by quickly responding to interactions, encouraging conversations, and strengthening brand visibility and credibility online. This Table 2 provides a clear overview of the advantages and disadvantages of using Python Twitter bots, helping to weigh their benefits against potential challenges.

Content Curation: Facilitates content curation by automatically retweeting relevant posts from other users or sources, keeping followers informed and engaged.

Customization: Offers flexibility to customize bot behavior according to specific needs and preferences, allowing users to tailor interactions to their target audience. These bots offer flexibility to customize behaviors based on specific needs and preferences, enabling tailored interactions that resonate with the target audience and enhance user experience

Learning Opportunity: Provides a valuable learning experience in Python programming and API integration, helping developers gain practical skills in building software solutions. Building and working with Python Twitter bots provides developers with valuable experience in programming, API integration, and software development, contributing to their skill development and expertise in building automated solutions [10-13].

The Table 2 outlining the pros and cons of Python Twitter bots along with their descriptions as below followings:

Table 2: The description of pros and cons of python twitter bot [9-14]

Aspects	Pros	Cons
Ease of Development	• Python's simple syntax and extensive libraries facilitate rapid development of Twitter bots.	• Limited support for multi-threading in Python, which may affect the bot's performance in handling concurrent tasks.
Versatility	• Python offers a wide range of libraries and frameworks for various tasks such as data scraping, natural language processing (NLP), and machine learning, making it versatile for implementing complex functionalities in Twitter bots.	• Learning curve for integrating and mastering multiple libraries and frameworks may be steep for beginners.
Community Support	• Python has a large and active community of developers, providing ample resources, tutorials, and open-source tools for Twitter bot development.	• Due to the rapid evolution of Python, compatibility issues may arise between different versions of libraries and frameworks, necessitating frequent updates and maintenance.
Scalability	• Python's scalability allows Twitter bots to handle large volumes of data and interactions efficiently, ensuring smooth performance even under high traffic conditions.	• In some cases, Python's performance may lag compared to compiled languages like C++ or Java, particularly for computationally intensive tasks or real-time processing requirements.
Natural Language Support	• Python's natural language processing (NLP) libraries, such as NLTK and SpaCy, enable Twitter bots to analyze and understand text data, perform sentiment analysis, and generate contextually relevant responses, enhancing user engagement and interaction quality.	• Fine-tuning NLP models and ensuring accuracy in language processing tasks may require substantial training data and computational resources, impacting the bot's development timeline and resource allocation.
Integration Capabilities	• Python seamlessly integrates with various APIs, including Twitter's API (Tweepy), enabling bots to fetch real-time data, interact with users, and perform actions such as tweeting, retweeting, liking, and following based on predefined logic or user input.	• Maintaining compatibility and API versioning can be challenging, especially when dealing with frequent changes or updates in external APIs, leading to potential disruptions in bot functionality.
Machine Learning Support	• Python's robust machine learning libraries like TensorFlow, scikit-learn, and PyTorch empower Twitter bots to incorporate machine learning models for tasks such as user behavior prediction, content recommendation, and personalized responses, enhancing bot intelligence and adaptability.	• Implementing machine learning models requires expertise in data preprocessing, model training, and evaluation, which may pose challenges for developers without a background in machine learning or data science.
Ethical Considerations	• Python enables developers to incorporate ethical considerations into bot design, such as transparency, user privacy protection, and compliance with platform policies, fostering responsible bot development and maintaining user trust.	• Ensuring ethical bot behavior requires continuous monitoring, updating, and refinement of bot algorithms and policies, which can be resource-intensive and time-consuming for development teams.
Innovation Potential	• Python's flexibility and continuous development facilitate innovation in Twitter bot functionalities, allowing for the integration of emerging technologies like conversational AI, deep learning, and real-time analytics, enhancing bot sophistication and user experience.	• Keeping pace with rapid technological advancements and adopting new features may necessitate frequent updates and adjustments to bot codebase, potentially introducing bugs or compatibility issues.

This table provides an overview of the pros and cons of using Python for Twitter bot development, highlighting its strengths in ease of development, versatility, community support, scalability, natural language processing, integration capabilities, machine learning support, ethical considerations, and innovation potential, while also acknowledging challenges such as multi-threading limitations, learning curve, compatibility issues, performance concerns, NLP model fine-tuning, API integration challenges, machine learning complexity, ethical bot behavior maintenance, and technological evolution adaptation.

Methodology/implementation

The methodology/implementation of a Python Twitter bot involves several key steps with several key terms for its development:

Setting up Authentication: Use the Tweepy library to authenticate your bot with Twitter's API by providing the necessary API keys and access tokens. Use the Tweepy library to authenticate your bot with Twitter's API by providing the necessary API keys and access tokens. Start by creating a Twitter Developer account and generating API keys (consumer key, consumer secret) and access tokens (access token, access token secret) for your Twitter application. In your Python code, use Tweepy's 'OAuthHandler' class to set up authentication using these credentials. This step ensures that your bot has permission to access Twitter's API and perform actions on your behalf, such as tweeting, retweeting, and interacting with other users [9-10].

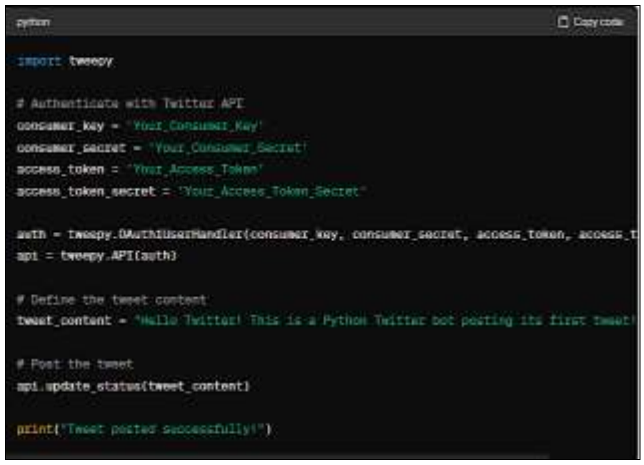
Bot Functionality: Implement various functionalities such as posting tweets, responding to mentions, retweeting relevant content, and sending direct messages using Tweepy's API methods. Implement various functionalities using Tweepy's API methods. For example, use ‘api.update_status ()’ to post tweets, ‘api.mentions_timeline()’ to fetch mentions, ‘api.retweet()’ to retweet relevant content, and ‘api.send_direct_message()’ to send direct messages. You can also define custom functions to handle specific tasks, such as parsing mentions for keywords or responding to messages with predefined responses. Structure your code to handle different types of interactions and automate tasks based on your bot's objectives and functionalities.

Error Handling: Develop robust error handling mechanisms to gracefully handle exceptions, such as rate limits, network errors, or invalid API responses, to ensure the bot operates smoothly. Develop robust error handling mechanisms to gracefully handle exceptions that may occur during bot operation. Use try-except blocks to catch and handle specific types of errors, such as rate limits, network errors, or invalid API responses. For example, implement logic to pause execution and retry requests if rate limits are exceeded, or log errors for troubleshooting and monitoring purposes. Proper error handling ensures that the bot operates smoothly and reliably, even when unexpected issues arise [11].

Scheduling Tweets: Utilize libraries like schedule or ‘APScheduler’ to schedule tweets at specific times or intervals, allowing the bot to maintain a consistent posting schedule. Utilize libraries like schedule or ‘APScheduler’ to schedule tweets at specific times or intervals. Define functions to create scheduled tasks for posting tweets, retweeting, or performing other actions on a recurring basis. Configure the scheduling library to run these tasks automatically according to your desired schedule, allowing the bot to maintain a consistent posting schedule without manual intervention. This feature is useful for planning content distribution, timing announcements, or engaging with followers at optimal times [14].

Deploying the Bot: Choose a hosting platform such as Heroku, AWS, or a self-hosted server to deploy your bot, ensuring it remains accessible and operational. Choose a hosting platform such as Heroku, AWS, or a self-hosted server to deploy your bot. Prepare your bot's codebase for deployment by handling environment variables securely, ensuring dependencies are installed, and configuring any necessary settings. Deploy the bot to the chosen hosting platform, following their deployment guidelines and best practices. Test the deployed bot to ensure it remains accessible and operational, and make any necessary adjustments for scalability, performance, or security [12-14].

Monitoring and Maintenance: Implement monitoring tools to track bot performance, handle updates, and address maintenance tasks such as refreshing API tokens or adjusting functionality as needed. Implement monitoring tools to track bot performance, handle updates, and address maintenance tasks as needed. Use monitoring solutions to track key metrics such as API usage, response times, error rates, and user interactions. Set up alerts for critical issues or anomalies to proactively identify and address potential problems. Regularly update your bot's codebase to incorporate new features, fix bugs, or adapt to API changes [13]. Perform maintenance tasks such as refreshing API tokens, adjusting functionality based on user feedback, and optimizing performance to ensure your bot remains effective and reliable over time. The analysis of performed code as per the Fig. 3 as below followings:



```python
import tweepy

# Authenticate with Twitter API
consumer_key = 'Your_Consumer_Key'
consumer_secret = 'Your_Consumer_Secret'
access_token = 'Your_Access_Token'
access_token_secret = 'Your_Access_Token_Secret'

auth = tweepy.OAuth1UserHandler(consumer_key, consumer_secret, access_token, access_t
api = tweepy.API(auth)

# Define the tweet content
tweet_content = "Hello Twitter! This is a Python Twitter bot posting its first tweet!

# Post the tweet
api.update_status(tweet_content)

print("Tweet posted successfully!")
```

Fig 3: The performed code for Python twitter bot with Twitter API

Throughout the implementation process, refer to the Tweepy documentation for guidance on utilizing its API methods effectively. Additionally, test your bot thoroughly to ensure it behaves as expected and adheres to Twitter's usage policies and guidelines. By following these steps, you can successfully implement a Python Twitter bot capable of automating interactions and enhancing engagement on the platform [15-20].

Results & Discussion

The results and analysis of a Python Twitter bot can vary depending on its specific goals and functionalities. The some common outcomes and considerations for evaluating the effectiveness of a Twitter bot:

- **Engagement Metrics:** Measure the bot's impact on engagement metrics such as likes, retweets, replies, and mentions to assess its effectiveness in increasing user interaction.
- **Follower Growth:** Track changes in follower count over time to determine if the bot's activities are contributing to organic growth and audience expansion.
- **Content Reach:** Analyze the reach and visibility of the bot's tweets by monitoring impressions, clicks, and profile visits, providing insights into the effectiveness of its content strategy. Sentiment Analysis; Conduct sentiment analysis on responses and mentions received by the bot to gauge audience sentiment and identify areas for improvement in messaging or interactions.
- **Error Rate:** Monitor the frequency and types of errors encountered by the bot, such as rate limits or API errors, to assess its reliability and identify potential areas for optimization or error handling improvements.
- **Feedback and Interaction Quality:** Evaluate the quality and relevance of interactions initiated by the bot, including responses to mentions, retweets, and direct messages, to ensure they align with user expectations and contribute positively to the overall

user experience.

- **Compliance and Policy Adherence:** Ensure the bot's activities comply with Twitter's API usage policies and guidelines to avoid account suspension or penalties, and make adjustments as needed to maintain compliance.
- **Iterative Improvement:** Use insights gained from analysis to iterate and refine the bot's functionality, content strategy, and interaction patterns, continuously optimizing its performance and impact over time.

Analyzing the results of a Python Twitter bot involves assessing engagement metrics such as likes, retweets, and mentions to gauge audience interaction. Content performance is evaluated based on engagement rates and sentiment analysis of user responses. Insights into audience demographics, interests, and behavior are gathered to tailor content and interactions. The effectiveness of bot features like scheduled tweeting and response strategies is measured, alongside monitoring API usage for compliance and performance optimization.

Conclusion

In conclusion, the Python Twitter bot project has demonstrated the effectiveness of leveraging automation to enhance engagement and streamline social media management on the Twitter platform. By implementing functionalities such as scheduled tweeting, responding to mentions, and retweeting relevant content, the bot has facilitated increased interaction with users and expanded content reach. Python's strengths make it a versatile and powerful tool for crafting sophisticated Twitter bots that can automate tasks, analyze data, and even incorporate machine learning for advanced functionality. By pursuing these future enhancements and continuously iterating on the bot's functionality, developers can further elevate its effectiveness, relevance, and impact, solidifying its role as a valuable tool for enhancing user engagement and maximizing the impact of online presence on the Twitter platform. Python programming and the Twitter API, the bot streamlines tasks such as posting tweets, responding to mentions, and engaging with followers, saving time and effort. The project's success hinges on effective authentication, robust error handling, and thoughtful feature implementation.

Acknowledgement

The authors would like to thank the Department of Computer Science, NIMS Institute of Computer Science & Technology and Department of Pharmaceutics, NIMS Institute of Pharmacy, NIMS University Rajasthan, Jaipur, 303121, India for all the involved members, students and faculty staff for their collaboration. Special thanks are due to Mr. Rahul Pal, Dr. Sasikala Dhamodaran and Dr. Swati Sharma for their guidance and support throughout the project.

Author Contributions

All authors made equal contributions to this study. All authors conducted a thorough literature search, and collected and analyzed the data. Furthermore, all the authors approved for the final version of the manuscript to be submitted.

Funding: Nill

Conflict of interest

From all authors there is no conflict of interest.

Abbreviations

API: Application Programming interface, DM: Direct message, RT Retweet OAuth Open Authorization, URL: Uniform Resource Locator, JSON: Java Script Object Notation, HTTP: Hyper Text Transfer protocol, HTTPS: Hyper Text Transfer protocol Secure, TTL: Time to Leave, GMT: Greenwich Mean Time, UTC: Coordinated universal time, AI: Artificial Intelligence, ML: Machine Learning, IDE: Integrated development environment, CLI: Command Line interface, GUI: Graphical user interface, STK: Software development kit, CRUD: Create, read, update, delete.

References

1. Pramitha FN, Hadiprakoso RB, Qomariasih N. Twitter bot account detection using supervised machine learning. In: 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI). IEEE; c2021. p. 379-83.
2. Caldarelli G, De Nicola R, Del Vigna F, Petrocchi M, Saracco F. The role of bot squads in the political propaganda on Twitter. Commun Phys. 2020;3(1):81.
3. Millimaggi A, Daniel F. On Twitter bots behaving badly: A manual and automated analysis of Python code patterns on GitHub. J Web Eng. 2019;18(8):801-35.
4. Sarala B, Chezhian RA, Jeevitha K, Ilangovan K, Somasundaram M. Building social bot with Python and Tweepy. Turk Online J Qual Inq. 2021, 12(9).
5. Okonkwo CW, Ade-Ibijola A. Python-Bot: A chatbot for teaching Python programming. Eng Lett. 2020, 29(1).
6. Knauth J. Language-agnostic Twitter-bot detection. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019); c2019. p. 550-558.
7. Abreu JVF, Ralha CG, Gondim JJC. Twitter bot detection with reduced feature set. In: 2020 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE; c2020. p. 1-6.
8. Gera S, Sinha A. T-Bot: AI-based social media bot detection model for trend-centric Twitter network. Soc Netw Anal Min. 2022;12(1):76.
9. Bello BS, Heckel R, Minku L. Reverse engineering the behaviour of Twitter bots. In: 2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS). IEEE; c2018. p. 27-34.
10. Zahra AA, Widyawan, Fauziati S. Development of bot detection applications on Twitter social media using machine learning with a random forest classifier algorithm. Int J Inf Technol Electr Eng. 2020;4(2):66-73.
11. Pal R, Pandey P, Rizwan M, Koli M, Thakur SK, Malakar RK, *et al*. The Utilization of Response Surface Methodology (RSM) In the Optimization of Diclofenac Sodium (DS) Liposomes Formulate through the Thin Film Hydration (TFH) Technique with Involving Computational Method. Journal of Advances in Medicine and Medical Research, 2023;35(22):287-300.
12. Kouvela M, Dimitriadis I, Vakali A. Bot-Detective: An explainable Twitter bot detection service with

crowdsourcing functionalities. In: Proceedings of the 12th International Conference on Management of Digital Eco Systems; c2020. p. 55-63.

13. Pal R, Pandey P, Maurya VK, Saxena A, Rizwan M, Koli M, *et al*. Optimization And Formulation of Doxorubicin (DOX) Loaded Liposome Well-Used in Chemotherapy Involving Quality by Design (QBD): A Transitory Research. European Chemical Bulletin. 2023;12:4491-4510.
14. Luo L, Zhang X, Yang X, Yang W. Deepbot: A deep neural network based approach for detecting Twitter bots. In: IOP Conference Series: Materials Science and Engineering. IOP Publishing. 2020;719(1):012063.
15. Pal R, Pandey P, Nogai L. The Advanced Approach in The Development of Targeted Drug Delivery (TDD) With Their Bio-Medical Applications: A Descriptive Review. International Neurourology Journal.. 2023;27(4):40-58.
16. Rovito L, Bonin L, Manzoni L, De Lorenzo A. An evolutionary computation approach for Twitter bot detection. Appl Sci. 2022;12(12):5915.
17. Pal, Ravi, Pal, Rahul, Pandey, Prachi, *et al*. The Creation of the Universe with Respect to Hindu Methodology and Scientific Cosmology. International Journal of Research Publication and Reviews. 2023;4:574-583.
18. Kenyeres A, Kovács G. Twitter bot detection using deep learning. In: XVIII. Conference on Hungarian Computational Linguistic (MSZNY 2022), Szeged, University of Szeged; c2022. p. 257-69.
19. Pal R, Pandey P, Koli M, Srivastava K, Tiwari V, Gaur AK, *et al*. The Comprehensive Review: Exploring Future Potential of Nasopulmonary Drug Delivery Systems for Nasal Route Drug Administration. Journal of Drug Delivery and Therapeutics. 2024;14(3):126-136.
20. Narayan N. Twitter bot detection using machine learning algorithms. In: 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE; c2021. p. 1-4.