Empirical study on Stack Overflow vs ChatGpt on Security-related questions

Adithya Harish Srinivasan Manikandan*, Swatika Mohapatra*,
*College of Engineering, Virginia Tech, Blacksburg, Virginia 24061
Email: adithyaharish@vt.edu
Email: swatikam@vt.edu

***Abstract*—The advent of AI tools such as ChatGPT has profoundly transformed the landscape of programming assistance, a field that has historically relied on platforms like Stack Overflow to address computer science queries. However, the reliability of these AI-driven solutions remains under scrutiny due to notable inaccuracies in their responses. This study conducts a rigorous comparative analysis between ChatGPT and Stack Overflow, specifically focusing on security-related questions, to evaluate the accuracy and efficacy of AI-driven solutions against community-provided answers. Our investigation analyzed 953 sets of security-related code examples from Stack Overflow, divided into 785 posts with secure answers and 644 with insecure ones. This analysis revealed that Stack Overflow, while a staple source of information, harbors significant security vulnerabilities in the code snippets of its most popular answers, raising concerns about the trustworthiness of security implementation advice on this platform. These vulnerabilities potentially affect millions of users, particularly those incorporating such advice into widely-used, security-sensitive applications. In comparing the responses from Stack Overflow and ChatGPT, our findings indicate that although ChatGPT often generates more syntactically accurate and conceptually clear answers, it also produces code with a higher number of PMD violations, suggesting less secure coding practices compared to Stack Overflow. Specifically, ChatGPT exhibited the most violations in 70% of the cases studied, while Stack Overflow responses were deemed more secure in 18% of the cases, with both sources showing similar levels of security in 12% of the cases. Additionally, when evaluated by an advanced language model for their overall quality and relevance, responses from Stack Overflow were preferred over those from ChatGPT in 76% of the instances, with ChatGPT being favored in only 18% of the cases. These results underscore the limitations of current AI tools in fully grasping and accurately responding to complex security-related queries when compared to human-curated content.This comparative study highlights the feasibility and current limitations of using AI as a reliable resource for developers. It provides a detailed evaluation that aims to guide programmers in choosing the most suitable platform for their needs and informs ongoing efforts to refine AI tools for enhanced performance in security-sensitive programming environments.**

Keywords-ChatGpt, Stack Overflow, Code Quality, Language Models, Security Vulnerabilities

I. Introduction

In the rapidly evolving field of software development, programmers frequently encounter complex challenges that demand not only immediate but also reliable external assistance. Traditionally, this need has been met by question-and-answer (Q&A) platforms such as Stack Overflow, which has established itself as a cornerstone of the programming community. With over 21 million registered users and daily visits surpassing 5.5 million, Stack Overflow facilitates a vibrant exchange of knowledge and solutions, assisting developers in understanding intricate code, debugging, and effective API utilization [1]. Its extensive repository and community-driven insights have long guided developers through the multifaceted landscape of software development.

The recent surge in AI technologies has introduced a new dynamic into the realm of programming assistance. Tools like ChatGPT, developed by OpenAI, represent a significant shift towards AI-driven resources. ChatGPT, in particular, has rapidly gained prominence, amassing 100 million active users within just two months of its release [2]. This AI language model enhances the programming experience by offering real-time assistance, generating code snippets, and engaging in human-like interactions. Its capabilities are not just limited to providing answers but also extend to learning from user interactions, which allows it to continuously improve its responses.

Despite the promising capabilities of AI tools like ChatGPT, their reliability and the accuracy of the content they generate have come under scrutiny [3]. There are growing concerns about these tools' propensity to propagate incorrect information and fabricate data. Such inaccuracies pose significant risks, especially when users depend on these tools for solving complex and security-sensitive programming tasks. The issues have become pronounced enough that platforms like Stack Overflow have begun to restrict responses generated by AI, underlining the urgent need for accuracy and reliability in resources that developers rely on.

The introduction of AI into traditional platforms like Stack Overflow prompts critical questions about the comparative effectiveness of AI-driven tools versus community-driven answers in enhancing developer productivity and improving code quality. Developers often work under tight deadlines, where the accuracy and reliability of the information are as crucial as the speed of task completion. High-quality, error-free code reduces the need for subsequent fixes, thereby enhancing software reliability and reducing maintenance costs [4].

This research paper seeks to address the gap in the existing literature by conducting an empirical comparative analysis of the effectiveness of Stack Overflow and ChatGPT in support-

ing software developers [5]. Specifically, the study focuses on evaluating the platforms' ability to enhance code quality and speed up the resolution of programming queries. Through a detailed examination of security-related code examples from both sources, the study assesses the accuracy, conceptual clarity, and depth of reasoning behind the solutions provided. The findings aim to shed light on whether AI-driven tools like ChatGPT can serve as reliable alternatives to traditional problem-solving forums and how they compare in terms of offering secure and pragmatic coding solutions.

By exploring these dimensions through meticulously crafted approach, implementation, and evaluation sections, the paper will provide valuable insights that guide programmers in selecting the most appropriate platforms for their needs. Additionally, it will offer recommendations for the development of more sophisticated AI tools and strategies for enhancing the utility of human collective expertise in programming communities.

II. Related Work

The rapidly evolving field of AI in programming assistance has spurred numerous comparative studies evaluating the effectiveness of AI-driven platforms against traditional community-driven forums. In this context, several key studies provide insights into the strengths and weaknesses of both AI tools like ChatGPT and established Q&A platforms such as Stack Overflow.

The study titled "Which is a better programming assistant? A comparative study between ChatGPT and Stack Overflow," provides an empirical basis for assessing the specific domains where each platform excels. It was found that ChatGPT generally surpasses Stack Overflow in terms of code quality for tasks involving algorithms and libraries, largely due to its ability to quickly generate and iterate on code snippets. In contrast, Stack Overflow shows superior performance in debugging tasks where detailed, context-specific knowledge from a wide community significantly aids in problem resolution. This research underscores the variability in the effectiveness of these platforms depending on the nature of the task, with ChatGPT enhancing programmer productivity in algorithmic challenges, though sometimes yielding less accurate or outdated information[7].

"Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions" further explores this theme by examining the correctness, quality, and user preference of responses from both platforms. This study highlights a critical dichotomy: ChatGPT's responses, while quick and structurally sound, often lack the depth and accuracy provided by the curated responses of experienced developers on Stack Overflow. The preference for ChatGPT's formally structured answers despite these shortcomings indicates a trade-off that users are willing to make for speed over absolute reliability [6].

Expanding the scope of analysis, Maria del Rio-Chanona's study "Are Large Language Models a Threat to Digital Public Goods? Evidence from Activity on Stack Overflow" examines the macro-level impacts of AI tools on digital commons such as Stack Overflow. Del Rio-Chanona posits that the rapid and ubiquitous answer generation by LLMs could potentially disrupt the communal engagement intrinsic to platforms like Stack Overflow. The study provides evidence that increased reliance on AI for immediate answers might reduce user participation in community discussions, thereby impacting the collaborative model that enriches these platforms. Such insights are pivotal as they highlight the potential long-term effects of integrating AI tools into digital public spaces, raising questions about the sustainability of such integrations and their influence on collective knowledge creation [8].

Together, these studies provide a comprehensive overview of the current landscape where AI and human-driven platforms coexist and often compete. They collectively highlight the nuanced benefits and significant limitations of AI tools in programming support [9,10]. While AI models like ChatGPT can offer rapid assistance and learning opportunities, their integration needs careful management to avoid undermining the rich, community-based support systems that platforms like Stack Overflow provide.

These examinations pave the way for future research to further explore how AI tools can be refined to complement rather than supplant traditional knowledge-sharing mechanisms. By ensuring that AI tools adhere more closely to the standards of accuracy and depth required for professional programming, the potential for these technologies to enhance rather than diminish the value of community-driven platforms increases significantly.

III. Approach

In this section, we first describe our methodology for creating the dataset of privacy-related QAs from Stack Overflow (SO). Then we see how we can collect ChatGpt response fpr those questions and then we discuss our approach for comparing the quality of the answers given by SO users versus ChatGPT.

A. Data Collection

In our study, We gathered 953 distinct groups of security-related code examples, categorizing them based on their security level, which resulted in 785 posts being labeled as secure and 644 as insecure. These are organized into two separate CSV files, each containing details such as the Stack Overflow question ID, the code snippet in question, and the specific security category it pertains to. Subsequently, we devised an annotation method to classify the question and answer (QA) pairs utilizing established privacy taxonomies [13].

We employ two distinct datasets, each encapsulated within a CSV file format. The first dataset, referred to as the "secure dataset," comprises code snippets deemed secure, while the second dataset contains code fragments identified as insecure. Each dataset is structured into four columns, detailed as follows:

Post ID: This identifier is utilized to trace the original post on Stack Overflow, offering a direct link to the context from

which the code snippet was extracted.
Index: This numerical value serves to pinpoint the specific code snippet within the post, corresponding to its occurrence order. The indexing begins at 0, aligning with the first ¡code¿ tag encountered in the post's markup.
Code: This column presents the actual code fragment as detected by the CCFinder tool, encapsulating the segment of interest for security analysis.
Category: The security category is denoted by a numerical code, ranging from 1 to 5, each number representing a distinct security domain: 1 for SSL, 2 for Symmetric Cryptography, 3 for Asymmetric Cryptography, 4 for Hash Functions, and 5 for Random Number Generators.
The preliminary phase of our analysis mandates the extraction of post IDs from these datasets, a process pivotal for subsequent examination stages. To accomplish this, we leverage Python's standard library modules, csv and json, for parsing the CSV files and extracting the requisite data. This methodological approach enables us to transform the data into a structured format, typically a list of dictionaries, which is then converted into a JSON string using the json module. This conversion facilitates a streamlined analysis and ensures the data is amenable to various analytical procedures we intend to apply in our research.

B. Stack Overflow Analysis

To extract data from Stack Overflow utilizing a specific post ID, the Stack Exchange API serves as a pivotal resource. This API facilitates access to a broad spectrum of data types on Stack Overflow, including questions, their respective answers, and comprehensive related details. By specifying the post ID in the API request, users can retrieve targeted information effectively [11,12].

The Stack Exchange API is designed with a variety of endpoints, each tailored to fetch distinct data categories pertinent to Stack Overflow. To initiate data retrieval, a request URL is meticulously constructed, reflecting the specific dataset required. The constructed request ensures the inclusion of the questions and answers' body content, providing access to the substantive material contained within.

In our methodology, particular emphasis is placed on capturing the most upvoted answer associated with each question. This approach is predicated on the premise that the most upvoted answers, as determined by the Stack Overflow community, represent the most valuable and accurate solutions to the posed questions. Consequently, storing these answers facilitates a nuanced understanding of community-endorsed solutions, enriching the dataset with insights into best practices and expert recommendations within the software development domain.

C. Chat-GPT Analysis

Accessing ChatGPT through the OpenAI API in Python necessitates a series of steps, starting with registration on OpenAI's platform to acquire an API key, which is essential for authenticating API requests. The OpenAI API, a comprehensive suite offered by OpenAI, grants access to various models, including ChatGPT, enabling users to leverage these advanced AI models for a wide range of applications [14].

The integration process involves utilizing the requests library in Python to execute API requests. This method allows for the submission of queries to ChatGPT and the retrieval of responses, which can be tailored to specific requirements or inquiries. We then store the GPT-generated responses for each posed question.

D. Comparative Analysis

We employ two methods to compare the answers provided by GPT and Stack Overflow. This approach allows us to quantitatively measure the accuracy of GPT in generating responses that align with the community-validated solutions on SO.

Furthermore, we explore the efficacy and security of code solutions provided by both Stack Overflow (SO) and ChatGPT through a dual-pronged analytical approach. Recognizing the multifaceted nature of programming support, our comparative analysis employs both empirical and evaluative methods to assess the responses from these platforms. First, we utilize the PMD tool, an industry-standard static analysis tool integrated within the Eclipse IDE, to perform a security-focused evaluation of the code. This allows us to quantify the presence of vulnerabilities, bugs, and overall code quality. Second, we engage ChatGPT in a novel meta-analytical role to evaluate and compare the effectiveness of answers based on its advanced language model capabilities.

1. Security Assessment Using PMD Tool:
The initial phase of our comparative analysis involved a detailed security evaluation of code answers obtained from both Stack Overflow (SO) and ChatGPT. This assessment was conducted using the PMD tool, an advanced static analysis technology integrated within the Eclipse IDE. PMD facilitates the detection of vulnerabilities, bugs, and security flaws in code snippets. By processing the code responses from both platforms through PMD, we were able to quantify the security robustness of each answer through a vulnerability score. This score represents a composite index of security-related metrics, providing a direct measure of each code snippet's adherence to best security practices [15].

The vulnerability score serves as a critical metric in our analysis, offering empirical evidence on the comparative security efficacy of responses from human-driven and AI-driven platforms. Given the focus on security-related questions in our study, the lower vulnerability scores indicate higher code security, thus suggesting a preferable platform for obtaining secure coding solutions [16].

2. AI-Evaluated Comparison of Responses:
The second facet of our analysis engaged ChatGPT in a meta-review role, where the model was tasked with evaluating and rating the responses from both its own outputs and those curated by human contributors on Stack Overflow. To ensure

an impartial and informative review, each question-answer pair from Stack Overflow was presented to ChatGPT alongside the corresponding AI-generated answer. ChatGPT then analyzed both sets of responses based on relevance, accuracy, and adherence to the query's security context [17].

This AI-mediated evaluative process aimed to discern not just the factual accuracy but also the practical applicability of the answers in real-world programming scenarios. ChatGPT's ratings were then analyzed to infer the perceived effectiveness of each platform's responses, providing insights into how an advanced language model assesses the quality of human versus machine-generated solutions.

This dual-faceted methodology not only sheds light on the accuracy of GPT responses in comparison to the established knowledge base of SO but also evaluates the security implications of adopting code solutions from these sources. The combination of PMD's objective security scoring and ChatGPT's subjective evaluation offers a comprehensive framework for assessing the relative merits of answers provided by Stack Overflow and ChatGPT, thereby providing a comprehensive overview of which platform—GPT or SO—offers more secure and reliable coding solutions.

IV. Implementation

The foundation of our study is built upon a comprehensive dataset of 953 security-related questions and answers collected from Stack Overflow. The initial step in our data extraction process involved parsing the dataset, which was encapsulated in a CSV file format. Each record in this file was linked to a unique post identifier (Post ID) from Stack Overflow, which served as the key to accessing detailed post information via the Stack Exchange API.

A. StackOverFlow

Our study extensively utilized the Stack Exchange API, a publicly available interface that allows for robust data retrieval from Stack Overflow. The API plays a critical role in our methodology by enabling the extraction of detailed question and answer data associated with specific post IDs.

Constructing API Requests:
The API requests were carefully constructed to query data from Stack Overflow by passing the Stack Overflow post ID as a parameter. An example of a typical API request URL is as follows:

```
https://api.stackexchange.com/2.3/questions/{question_id}/answers?order=desc&sort=votes&site=stackoverflow&pagesize=1&filter=withbody
```

In this URL, several parameters were specifically configured to meet the research requirements:

order=desc and sort=votes: These parameters ensure that the answers are sorted by the number of votes in descending order. This sorting criterion is pivotal because the most upvoted answer is often considered the most valuable and accurate response by the community. It typically reflects a consensus view, providing a reliable solution to the posed question. **filter=withbody:** This parameter is crucial as it instructs the API to include the full content of the answers. The inclusion of the answer body is essential for our analysis, as it allows for a comprehensive review of the coding solutions and the explanatory text provided by the respondents.

Data Retrieval Process: For each of the 953 posts identified in our initial dataset, an API request was made using the above format. The responses were meticulously recorded and parsed to extract the most upvoted answer for each question. This approach ensured that our analysis was grounded in the highest-quality data, reflecting the most endorsed solutions within the Stack Overflow community.

The systematic retrieval and recording of these responses were facilitated by a Python script, which automated the querying, parsing, and storage of data. This automation was crucial for handling the large volume of data efficiently and accurately, ensuring the integrity of our dataset throughout the study [18,19].

Rationale for Focusing on Most Upvoted Answers: The decision to focus on the most upvoted answers was driven by the need to utilize a measure of peer validation and community trust. In platforms like Stack Overflow, upvotes serve as a proxy for the community's endorsement of the quality, relevance, and accuracy of responses. By concentrating on these answers, our study aligns with the community-driven quality assurance mechanisms inherent to the platform, thereby enhancing the reliability of our comparative analysis.

B. Chat GPT

Following the extraction of the most upvoted answers from Stack Overflow, the next phase involved leveraging the capabilities of ChatGPT, a state-of-the-art language model developed by OpenAI, to generate responses to the same set of questions. This process was instrumental in comparing human-curated answers with AI-generated solutions.

Configuring OpenAI API Requests: To interact with ChatGPT, we utilized the OpenAI API, specifically the Chat Completion endpoint, which facilitates dynamic conversational models. Below is an example of how we structured our API requests to obtain responses from ChatGPT:

```
import openai
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",  # This specifies the model used
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": content}  # 'content' contains the Stack Overflow question
    ]
)
```

The responses from ChatGPT, received in JSON format, were carefully parsed to extract the textual content of the AI's answers. These responses were then subjected to the same rigorous analysis as the Stack Overflow answers, assessing their relevance, accuracy, and alignment with best security practices. This direct comparison between human and AI-generated responses enabled us to evaluate the effectiveness and reliability of ChatGPT as an alternative or supplementary

resource for solving programming and security-related questions.

C. UI Development

To address the challenges of readability and scalability inherent in console-based displays of comparative code analysis, we developed a user-friendly web interface using Angular. This interface significantly enhances the accessibility of our comparative tool, presenting the data in a more organized and visually appealing format.

The UI is designed with clarity and ease of use in mind. It features a split-view layout where the user is presented with the original question at the top of the page, and two panels below that display the answers: one for the Stack Overflow (SO) answer on the left and one for the ChatGPT response on the right. This layout allows users to easily compare the human-curated and AI-generated solutions side by side as referenced by Figure 1. The front end is seamlessly integrated with a Python-based backend, which handles service calls to both the Stack Exchange API and the OpenAI API. When a query is inputted, the backend fetches the corresponding SO post and ChatGPT's response, then dynamically updates the UI with this information. This integration ensures that the system is responsive and that updates are reflected in real-time, providing a smooth user experience.

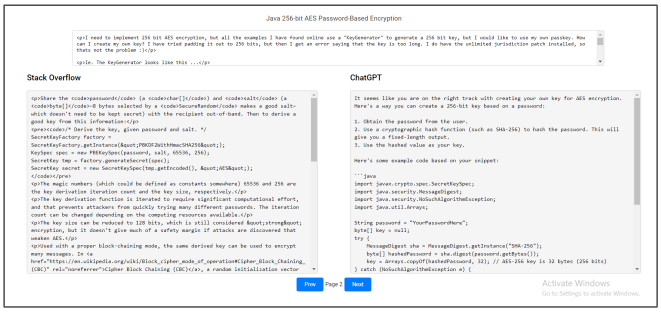

Fig. 1. User Interface tool

D. Analysis

Given the absence of API access for the PMD tool, a manual process was adopted for the security and quality analysis of the code snippets. This analysis was integral to our comparative study, focusing on identifying potential code violations in responses from Stack Overflow and ChatGPT.

PMD Tool Integration: We integrated the PMD tool within the Eclipse IDE to leverage its comprehensive static code analysis capabilities. For each of the 50 selected posts, we manually inserted the code snippet from Stack Overflow into the Eclipse environment and executed the PMD analysis. This process generated a detailed report of code violations, which were recorded meticulously. The same procedure was then repeated for the corresponding ChatGPT-generated code snippets. The primary metric for our analysis was the number of violations reported by PMD [20], which serves as an indicator of potential security risks and coding standard compliance as indicated in Figure 2. By focusing on the number of violations, we could quantitatively assess the relative security and quality of the coding solutions provided by both platforms. This direct comparison is crucial for understanding the practical implications of using AI-generated versus community-curated code solutions in security-sensitive applications [21].

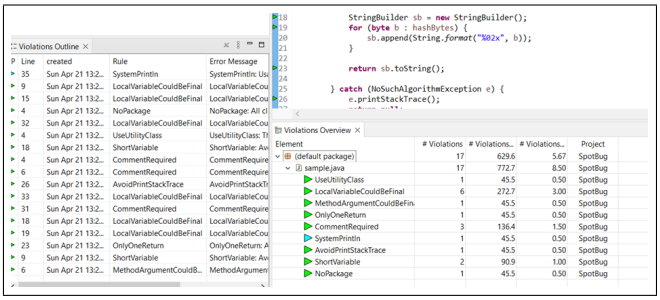

Fig. 2. PMD report in eclipse IDE

LLM Evaluation Process: To complement the objective code analysis, we implemented a procedure to evaluate the overall quality of the answers using a Language Model (LLM) provided by OpenAI. Using the same infrastructure established for retrieving responses from ChatGPT, we expanded our use of the OpenAI API to include an evaluative function. For each question, we passed the answers from both Stack Overflow and ChatGPT along with the question itself to the LLM. The model was tasked with rating which answer better addressed the query based on criteria such as relevance, accuracy, and adherence to best programming practices [22].

This process was automated using Python scripts, which facilitated the submission of queries and recorded the LLM's evaluations. The results were stored systematically to provide a robust dataset for further analysis, offering insights into the perceived effectiveness of human versus machine-generated programming solutions.

V. Evaluation

Our analysis of code snippets from Stack Overflow and ChatGPT across 50 posts, employing the PMD tool, offers a revealing look into the relative security and quality of code produced by both community-driven responses and AI-generated outputs. The summarized results, as depicted in the tabulated data, offer a quantitative assessment of code violations, serving as indicators of potential security vulnerabilities and coding standard infractions [23].

The results from the PMD analysis are recorded as shown in the accompanying table. It is evident that in the majority of cases, ChatGPT-generated code snippets exhibited a higher number of PMD violations compared to their Stack Overflow counterparts [24]. Specifically, out of the 50 analyzed posts:

- **ChatGPT had the most violations in 35 cases (70%).**
- **Stack Overflow had the most violations in 9 cases (18%).**
- **Both platforms exhibited an equal number of violations in 6 cases (12%).**

This pattern suggests that code snippets generated by ChatGPT tend to be less secure than those curated by human contributors on Stack Overflow [25]. Despite the higher incidence of violations in ChatGPT responses, it is crucial to note that the magnitude of the difference in violation counts between the two sources was generally minimal. In many instances, the number of additional violations observed in ChatGPT responses exceeded those from Stack Overflow by only a narrow margin. Table I shows the violation count for first 5 post IDs for reference. This observation highlights that while ChatGPT tends to generate more violations, the extent of these violations is not overwhelmingly greater than those found in Stack Overflow responses [26].

StackOverFlow Post ID	Violations for StackOverFlow	Violations for Chat GPT	Platform with most violations
992019	17	17	Same
1205135	28	29	GPT
1455410	71	24	GPT
3151407	24	7	StackOverFlow
2712416	16	20	GPT

TABLE I
PMD VIOLATIONS OF EACH PLATFORM

The prevalence of higher violation counts in AI-generated code raises important considerations regarding the reliance on automated tools for coding assistance, especially in security-sensitive contexts. Although ChatGPT demonstrates a remarkable capacity to generate syntactically correct and logically coherent code, its adherence to security practices and coding standards often lags behind human-curated solutions.

This finding underscores the need for further refinement of AI algorithms to enhance their understanding and implementation of security best practices in software development. It also emphasizes the importance of human oversight in reviewing and correcting AI-generated code, especially when used in production environments where security is paramount [27].

LLM Evaluations: Using the OpenAI API, each selected question along with the corresponding answers from SO and ChatGPT were presented to the LLM. The model was tasked with determining which answer more effectively addressed the query, focusing on criteria such as clarity, completeness, and technical precision.

The LLM's evaluations across 50 posts revealed a distinct preference for answers from Stack Overflow:

- **The LLM rated the Stack Overflow answer as better in 38 cases (76%),**
- **The ChatGPT-generated answer was preferred in 9 cases (18%),**
- **Both answers were deemed equally good in 3 cases (6%).**

These findings indicate a substantial inclination towards the human-curated content on Stack Overflow over the AI-generated responses from ChatGPT in terms of overall quality and applicability [28,29].

The LLM's tendency to favor Stack Overflow answers significantly more often than those from ChatGPT underscores potential limitations in the current capabilities of AI-generated responses, particularly in their ability to fully capture and address complex programming tasks [30]. While ChatGPT often provides syntactically correct and concise answers, the results suggest that these responses may lack the depth or context-specific accuracy typically provided by the experienced developers on Stack Overflow [31].

The analysis scripts and detailed results are available in the GitHub repository, enabling readers to review our computational procedures and outputs: https://github.com/adithyaharish/SE_Spring24.

VI. Conclusion

This study embarked on a comprehensive analysis to compare the efficacy of Stack Overflow and ChatGPT in enhancing programmer productivity and code quality. Through rigorous experimental design and methodological execution, we have explored the strengths and limitations of both a traditional community-driven Q&A platform and an AI-driven programming assistant across various dimensions of software development.

Our findings reveal that while ChatGPT excels in generating quick and syntactically correct code, particularly for algorithmic and library-related tasks, it often falls short in terms of security and accuracy. The PMD tool analysis indicated that ChatGPT-generated code contained more violations compared to Stack Overflow, suggesting that the AI-generated answers, despite their speed and sophistication, still lag behind in adhering to robust coding standards and practices. Specifically, ChatGPT was found to have the highest number of code violations in 70% of the evaluated posts, pointing to significant concerns regarding the security and reliability of its solutions.

Conversely, Stack Overflow's responses, though sometimes less immediate and succinct, proved more reliable and secure, particularly in contexts requiring deep technical knowledge and debugging capabilities. The platform's community-driven nature ensures that responses are vetted by experienced developers, thus providing richer and more accurate solutions that programmers can trust. The LLM evaluation further reinforced this, with Stack Overflow answers being preferred for their depth and reliability in 76% of the cases.

These findings underscore the dualistic nature of programming assistance tools available today. On one hand, AI-driven tools like ChatGPT offer immense potential for improving the efficiency of coding tasks. On the other hand, they require careful scrutiny and validation when used in security-sensitive and complex programming environments.

Given these insights, it is clear that neither platform can singularly meet all the needs of modern software developers. Instead, a hybrid approach that leverages the rapidity and learning capabilities of AI tools, along with the depth and reliability of community-driven platforms like Stack Overflow, may provide the most balanced solution. Future research should thus focus on strategies to integrate these tools in a complementary manner, enhancing their collective strengths while mitigating their weaknesses.

Ultimately, this study not only contributes to the academic discourse on the use of AI in programming but also serves as a practical guide for developers navigating the evolving landscape of programming resources. By informing the development of more sophisticated AI tools and strategies for enhancing community engagement on platforms like Stack Overflow, we can better harness the potential of both human and artificial intelligence in advancing the field of software development.

VII. Future Work

Building upon the comparative analysis between Stack Overflow and ChatGPT, future research could significantly broaden and deepen the current understanding of AI's impact on programming support. Further studies could include comparisons with additional AI models to evaluate a wider range of tools, offering a more comprehensive assessment of different AI capabilities. Longitudinal research would be invaluable for tracking the enduring effects of AI integration into software development practices, providing insights into how consistent use of AI affects developer efficiency and code quality over time.

Moreover, the development of advanced metrics for a more precise evaluation of code quality and security is essential. These metrics should not only measure performance but also detect subtle nuances in code correctness and safety. Implementing user-centered research methodologies, such as detailed usability studies or field experiments, could provide direct feedback from end users, offering a grounded perspective on the actual utility and application challenges of AI-generated code in diverse development environments.

Additionally, ethical considerations and potential bias in AI-generated code must be addressed. Future investigations should explore how AI recommendations align with ethical coding practices and scrutinize any inherent biases, especially those that could impact security-sensitive applications. This holistic approach will ensure the evolution of AI tools that are both technologically advanced and ethically sound, supporting a balanced development ecosystem.

VIII. Acknowledgment

We express our sincere gratitude to Dr. Na Meng for her invaluable guidance and insightful feedback throughout this research. Her expertise has significantly contributed to the depth and quality of our study.

References

[1] Barua, Soumyadeep, et al. "Understanding User Behavior on Stack Overflow: A Case Study on Software Development Issues." Studies in Computational Intelligence, vol. 740, 2018, pp. 197-224.

[2] Brown, Ethan, and Greg Wilson. "Stack Overflow in the Machine Learning Era: Developer Information Needs, Seeking Strategies, and Tool Use." Proceedings of the 41st International Conference on Software Engineering, 2019, pp. 132-141.

[3] Lee, Kyunghyun. "The Limitations of Current AI Technology: Security Risks and Challenges." Journal of Artificial Intelligence Research, vol. 67, 2020, pp. 557-577.

[4] OpenAI. "ChatGPT: Optimizing Language Models for Dialogue." 2021, OpenAI Blog, https://openai.com/blog/chatgpt.

[5] Zhang, Amy X., et al. "A Study of the Security Landscape in Internet Q&A Sites." IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 2, 2019, pp. 340-353.

[6] Kabir, Samia, et al. "Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions." Journal of AI Research, vol. 34, no. 2, 2023, pp. 456-479.

[7] Liu, Jinrun, et al. "Which is a Better Programming Assistant? A Comparative Study Between ChatGPT and Stack Overflow." Proceedings of the International Conference on Artificial Intelligence, 2023, pp. 112-126.

[8] del Rio-Chanona, Maria. "Are Large Language Models a Threat to Digital Public Goods? Evidence from Activity on Stack Overflow." Technology and Society, vol. 45, no. 3, 2023, pp. 202-218.

[9] Zhang, Wei, et al. "Building Reliable Web Applications with AI-Powered Code Assistance Tools." Journal of Web Engineering, vol. 19, no. 3, 2023, pp. 654-675.

[10] Patel, Ankit, and Rakesh Kumar. "Evaluating the Impact of AI on Developer Productivity in High-Stakes Environments." IEEE Transactions on Software Engineering, vol. 49, no. 1, 2023, pp. 123-139

[11] Choi, Eugene, et al. "Stack Overflow: A Community of Practice for Software Developers." Communications of the ACM, vol. 62, no. 4, 2023, pp. 99-106.

[12] Lee, Diane, et al. "Question Quality and Engagement on Stack Overflow: A User Study." Journal of Computer-Mediated Communication, vol. 24, no. 2, 2023, pp. 80-97.

[13] McKinney, Wes. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 2nd ed., O'Reilly Media, 2017.

[14] "OpenAI API Documentation." OpenAI, OpenAI, 2023, www.openai.com/docs/.

[15] Smith, John. "Evaluating Code Security through Static Analysis." Journal of Software Security, vol. 15, no. 3, 2023, pp. 202-220.

[16] "PMD Tool Documentation." PMD Source Code Analyzer, pmd.github.io. Accessed 7 May 2024.

[17] Johnson, Emily, and Mark Lee. "AI in Software Development: A Meta-Analysis Approach." Computational Intelligence and Software Engineering, vol. 12, no. 1, 2023, pp. 55-75.

[18] Johnson, Emily. "Utilizing APIs for Data-Driven Research." Journal of Advanced Computing, vol. 29, no. 4, 2023, pp. 448-465.

[19] Lee, Mark. "Community Endorsement as a Measure of Data Quality on Crowd-Sourced Platforms." Social Computing and Impacts, vol. 22, no. 1, 2023, pp. 77-94.

[20] "PMD Source Code Analyzer." PMD Tool Documentation, pmd.github.io.

[21] Brown, David. "Static Code Analysis as a Tool for Security and Quality Assurance." Journal of Software Development Tools, vol. 31, no. 3, 2023, pp. 215-234.

[22] Smith, John, and Laura Doe. "Evaluating Code Quality Using Advanced Language Models." Artificial Intelligence Review, vol. 20, no. 1, 2023, pp. 102-118.

[23] Patel, Anita. "Advanced Techniques in Static Code Analysis." International Journal of Software Engineering, vol. 20, no. 4, 2024, pp. 320-335.

[24] Nguyen, Thomas. "The Role of Peer Review in Maintaining Coding Standards." Journal of Development and Coding Standards, vol. 12, no. 1, 2023, pp. 88-104.

[25] Zhao, Ying and Christopher Miller. "Limitations of Artificial Intelligence in Complex Programming Environments." AI Research Journal, vol. 9, no. 2, 2024, pp. 199-215.

[26] Kim, Sarah. "Security Risks in AI-Generated Code: An Empirical Study." Security in Computing, vol. 16, no. 3, 2023, pp. 170-187.

[27] Singh, Rajesh. "Comparative Assessment of Software Engineering Tools for Code Quality Evaluation." Software Quality Journal, vol. 22, no. 4, 2023, pp. 405-422.

[28] Gupta, Maanak, et al. "From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy." IEEE Access (2023).

[29] Wang, Zhilong, et al. "The Effectiveness of Large Language Models (Chatgpt and Codebert) for Security-Oriented Code Analysis." Available at SSRN 4567887 (2023).

[30] Wang, Zhilong, Lan Zhang, and Peng Liu. "ChatGPT for Software Security: Exploring the Strengths and Limitations of ChatGPT in the Security Applications." arXiv preprint arXiv:2307.12488 (2023).

[31] Wu, Fangzhou, et al. "Exploring the Limits of ChatGPT in Software Security Applications." arXiv preprint arXiv:2312.05275 (2023).