Benchmarking Large Language Models for Safe Software Development Advice

Robert Sloane-Anderson¹ and Harvey Price¹

¹AInergetic

June 07, 2024

Abstract

Secure software development is essential for safeguarding systems against an increasing array of cyber threats. Evaluating the capabilities of advanced language models in providing secure coding advice is crucial for understanding their potential as reliable tools for developers. The study benchmarks ChatGPT and Google Gemini across metrics such as accuracy, relevance, comprehensiveness, and adherence to best practices, offering a detailed comparative analysis of their performance. Google Gemini demonstrated superior accuracy and adherence to secure coding standards, while ChatGPT excelled in generating contextually relevant and comprehensive responses. The findings highlight the strengths and weaknesses of each model, providing insights into their application in secure software development. The study’s results inform developers on leveraging these models to enhance their coding practices, contributing to the creation of more secure and resilient software systems.
Benchmarking Large Language Models for Safe Software Development Advice

Robert Sloane-Anderson*, and Harvey Price

Abstract—Secure software development is essential for safeguarding systems against an increasing array of cyber threats. Evaluating the capabilities of advanced language models in providing secure coding advice is crucial for understanding their potential as reliable tools for developers. The study benchmarks ChatGPT and Google Gemini across metrics such as accuracy, relevance, comprehensiveness, and adherence to best practices, offering a detailed comparative analysis of their performance. Google Gemini demonstrated superior accuracy and adherence to secure coding standards, while ChatGPT excelled in generating contextually relevant and comprehensive responses. The findings highlight the strengths and weaknesses of each model, providing insights into their application in secure software development. The study’s results inform developers on leveraging these models to enhance their coding practices, contributing to the creation of more secure and resilient software systems.

Index Terms—cybersecurity, large language models, secure coding, benchmarking, artificial intelligence, software development

I. INTRODUCTION

The ever-increasing complexity of software systems, coupled with the expanding landscape of cybersecurity threats, has asked for the paramount importance of secure software development. Ensuring the integrity, confidentiality, and availability of software systems is essential, as vulnerabilities can lead to significant financial, operational, and reputational damage. The adoption of secure software development practices is not merely a recommendation but a necessity in today’s digital age, where the proliferation of cyber-attacks poses substantial risks to organizations and individuals alike. The implementation of robust security measures throughout the software development lifecycle is critical to mitigating these risks and ensuring the development of resilient software systems.

Large Language Models (LLMs) have emerged as powerful tools capable of generating human-like text based on given inputs. Their applications span various domains, including natural language processing, customer support, content generation, and more recently, providing advice and guidelines for software development. The ability of LLMs to understand and generate contextually relevant and coherent text positions them as valuable resources for developers seeking to enhance their coding practices and ensure adherence to security standards. However, the efficacy and reliability of the advice generated by LLMs in the context of secure software development require thorough evaluation to determine their suitability and effectiveness in addressing security concerns.

The present study aims to conduct a comprehensive benchmarking of ChatGPT and Google Gemini, focusing on their capability to provide safe and accurate advice to software developers. This evaluation is driven by the need to ascertain whether LLMs can be reliably utilized as advisory tools in the realm of secure software development. Given the critical nature of security in software systems, it is essential to establish the validity and reliability of the recommendations generated by LLMs, ensuring they align with established best practices and standards in the field.

Secure software development practices encompass a wide array of methodologies, techniques, and guidelines designed to identify and mitigate potential security vulnerabilities during the software development lifecycle. These practices include, but are not limited to, secure coding standards, threat modeling, static and dynamic analysis, code reviews, and security testing. Organizations such as the Open Web Application Security Project (OWASP) and the Software Engineering Institute (SEI) have developed comprehensive frameworks and guidelines to assist developers in incorporating security considerations into their development processes. The adoption of these practices is crucial in preventing security breaches and ensuring the development of robust and secure software systems.

Large Language Models, exemplified by ChatGPT and Google Gemini, represent a significant advancement in the field of artificial intelligence and natural language processing. These models are trained on vast datasets and employ sophisticated algorithms to generate text that is contextually relevant and coherent. The potential applications of LLMs are diverse, ranging from automating customer support to generating creative content and providing educational resources. In the context of software development, LLMs have been leveraged to offer coding suggestions, debug assistance, and adherence to best practices. However, the application of LLMs in providing secure software development advice necessitates a rigorous assessment to ensure the accuracy and reliability of their recommendations.

The primary objective of this study is to evaluate the performance of ChatGPT and Google Gemini in providing safe and reliable advice on secure software development practices. By systematically benchmarking these LLMs, the study aims to determine their effectiveness in addressing common security concerns encountered by software developers. The evaluation will focus on key aspects such as the accuracy, relevance, and comprehensiveness of the advice generated by the LLMs, as
well as their adherence to established secure coding standards and guidelines. The ultimate goal is to provide insights into the potential and limitations of LLMs as advisory tools in the field of secure software development, thereby informing their future development and deployment in real-world scenarios.

This article:
1) Conducts a comprehensive benchmarking of ChatGPT and Google Gemini, focusing on their ability to provide secure software development advice.
2) Defines and utilizes precise evaluation metrics such as accuracy, relevance, comprehensiveness, and adherence to best practices.
3) Provides a detailed comparative analysis of the strengths and weaknesses of each model, offering valuable insights for developers.
4) Suggests practical implications and recommendations for the integration of LLMs into secure software development workflows.

II. RELATED STUDIES

The topic of secure software development has witnessed significant advancements, focusing on methodologies and tools designed to enhance the security posture of software systems. An extensive range of machine learning techniques, including supervised, unsupervised, and reinforcement learning, has been employed to detect and mitigate vulnerabilities within codebases [1], [2]. The application of static analysis tools facilitated the early detection of security flaws by analyzing the code without executing it, thereby preventing many potential exploits during the development phase [3]–[5]. Dynamic analysis techniques, which involve executing the code and monitoring its behavior, offered additional insights into runtime vulnerabilities that static analysis alone could not uncover [6], [7]. Secure coding practices, such as input validation, output encoding, and proper error handling, were emphasized to prevent common security issues like SQL injection and cross-site scripting [8], [9]. The integration of threat modeling into the development lifecycle allowed for the identification and mitigation of potential threats early in the design phase, significantly reducing the risk of security breaches [10], [11]. Automated tools for code reviews and security testing improved the efficiency and effectiveness of the secure development process, enabling the identification of vulnerabilities that manual reviews might miss [12], [13]. The adoption of continuous integration and continuous deployment (CI/CD) pipelines with embedded security checks ensured that security measures were consistently applied throughout the development lifecycle, maintaining a high level of security hygiene [14], [15]. The implementation of security training programs for developers enhanced their awareness and understanding of secure coding practices, contributing to the overall security of the software produced [16]. The use of formal methods and mathematical proofs in software verification provided a high level of assurance in the correctness and security of critical software systems [17]. The emphasis on building security into the software development lifecycle (SDLC) from the outset, rather than addressing it as an afterthought, resulted in more resilient and secure software products [18].

Large language models are often subject to rigorous benchmarking to evaluate their performance across a variety of tasks [19], [20]. The ability of LLMs to understand and generate human-like text enabled their application in diverse domains, including customer support, content generation, and educational tools [21], [22]. Benchmarks such as the General Language Understanding Evaluation (GLUE) and the SuperGLUE provided comprehensive frameworks for assessing the performance of LLMs on a wide range of natural language understanding tasks [19], [23]. The incorporation of context-aware mechanisms in LLMs improved their ability to generate contextually relevant and coherent text, enhancing their utility in real-world applications [24]. The evaluation of LLMs on zero-shot, one-shot, and few-shot learning tasks demonstrated their capacity to perform tasks with little to no task-specific training, showcasing their adaptability and generalization capabilities [25]. The use of adversarial testing revealed the robustness and vulnerabilities of LLMs, identifying areas where they excelled and where improvements were needed [26], [27]. The assessment of ethical considerations, such as bias and fairness, in LLM outputs highlighted the importance of developing models that produce unbiased and equitable results [28]. The comparison of LLMs with human performance on various benchmarks provided insights into their strengths and limitations, guiding future research and development efforts [29], [30]. The exploration of fine-tuning techniques for LLMs allowed for the customization of models to specific tasks or domains, enhancing their performance and relevance [31].

The development of evaluation metrics for LLM-generated text, including measures of fluency, coherence, and factual accuracy, enabled more precise assessments of model quality [32]. The application of LLMs in knowledge-intensive tasks, such as question answering and summarization, showcased their potential in providing valuable insights and information retrieval capabilities [33]. The ongoing improvements in model architectures and training techniques continued to push the boundaries of what LLMs could achieve, driving advancements in artificial intelligence and natural language processing [34], [35].

III. METHODOLOGY

The methodology employed in this study is designed to provide a rigorous and comprehensive evaluation of ChatGPT and Google Gemini in the context of delivering secure software development advice. The multifaceted approach involves the careful selection of benchmarks, the definition of precise evaluation criteria, the meticulous design of prompts, consistent model querying, the development of an automated analysis pipeline, and a thorough comparison and benchmarking process. Each component of the methodology is meticulously planned to ensure the reliability and validity of the findings.

A. Selection of Benchmarks

The selection of a benchmark is a critical step in the research methodology, aiming to ensure the evaluation is grounded in established and widely recognized standards for secure software development. The chosen benchmark must be
relevant to current secure coding practices, comprehensive in addressing a wide range of security issues, and widely accepted within the professional and academic communities. For this study, the OWASP Secure Coding Practices were selected as the primary benchmark. This selection was made due to OWASP’s extensive guidelines that cover a broad spectrum of security concerns and their wide acceptance within the software development community.

The OWASP Secure Coding Practices provide detailed guidelines for mitigating common security vulnerabilities. These guidelines encompass various aspects of software security, including input validation, authentication, session management, access control, cryptographic practices, error handling, and logging. To provide a clear overview of the OWASP Secure Coding Practices, table I was created to outline the key areas addressed by these guidelines.

The OWASP Secure Coding Practices were chosen for their comprehensive coverage of security issues that are critical to secure software development. By using this benchmark, the study aims to evaluate the ability of the LLMs to provide advice that aligns with widely accepted best practices, ensuring the relevance and applicability of the findings. This holistic approach allows for a thorough assessment of the advice generated by the LLMs, ensuring it meets the rigorous standards set by the OWASP guidelines.

B. Definition of Evaluation Criteria

The evaluation criteria were defined to objectively assess the quality and safety of the advice provided by ChatGPT and Google Gemini. The selected metrics encompass multiple dimensions to ensure a comprehensive assessment of the LLM responses. These metrics include accuracy, which measures the correctness of the advice; relevance, assessing how well the advice addresses the given prompt; comprehensiveness, evaluating the extent to which the advice covers all necessary aspects of the security issue; and adherence to best practices, ensuring the advice aligns with established secure coding standards.

The rationale for choosing these metrics lies in their ability to provide a multidimensional assessment of the LLM responses, capturing not only the factual correctness but also the practical applicability and completeness of the advice. Accuracy ensures that the advice given is factually sound and reliable, which is fundamental to providing trustworthy guidance. Relevance ensures that the responses are tailored to the specific security issues presented in the prompts, making the advice contextually appropriate and useful. Comprehensiveness guarantees that the advice covers all essential aspects of the security issue, providing a holistic approach to addressing the problem. Adherence to best practices ensures that the advice aligns with established standards, providing developers with guidance that is both practical and aligned with widely accepted secure coding practices. By employing these evaluation criteria, the study aims to provide a thorough and balanced assessment of the LLMs’ capabilities in delivering secure software development advice.

C. Prompt Design, Model Querying, and Automated Analysis

The process of prompt design, model querying, and automated analysis was integrated to ensure a comprehensive and efficient evaluation of ChatGPT and Google Gemini. A diverse and representative set of prompts reflecting real-world scenarios in secure software development was created. Prompts covered topics such as input validation, authentication mechanisms, cryptographic practices, error handling, and secure coding guidelines. Each prompt was formulated to elicit detailed and contextually relevant responses from the LLMs, capturing their ability to address complex security issues effectively.

The querying procedure was meticulously designed to ensure consistency and fairness. Both models were queried using the same set of prompts, with responses collected in a controlled environment to avoid biases or inconsistencies. Multiple iterations accounted for variations in responses, ensuring a robust assessment of performance. An automated analysis pipeline, incorporating natural language processing (NLP) algorithms and rule-based systems, was developed to systematically evaluate the LLM responses based on predefined criteria. This pipeline enabled efficient processing of large data volumes, ensuring objectivity and consistency while minimizing human bias and error.

D. Comparison and Benchmarking

The methods used to compare the performance of ChatGPT and Google Gemini involved sophisticated statistical analysis techniques to quantify the differences and similarities in their responses. Techniques such as t-tests and ANOVA were employed to determine the statistical significance of the observed differences in performance metrics. Additionally,
TABLE I
OWASP Secure Coding Practices Overview

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Validation</td>
<td>Ensure all inputs are validated, sanitized, and verified to prevent injection attacks.</td>
</tr>
<tr>
<td>Authentication</td>
<td>Implement robust authentication mechanisms, including multi-factor authentication and secure password storage.</td>
</tr>
<tr>
<td>Session Management</td>
<td>Use secure session management practices to prevent session hijacking and fixation.</td>
</tr>
<tr>
<td>Access Control</td>
<td>Enforce proper access control mechanisms to ensure that users can only access resources they are authorized to.</td>
</tr>
<tr>
<td>Cryptographic Practices</td>
<td>Use strong cryptographic algorithms and protocols to protect data at rest and in transit.</td>
</tr>
<tr>
<td>Error Handling</td>
<td>Implement secure error handling to prevent leakage of sensitive information through error messages.</td>
</tr>
<tr>
<td>Logging</td>
<td>Ensure logging of security-relevant events for monitoring and auditing purposes.</td>
</tr>
</tbody>
</table>

TABLE II
Evaluation Criteria for Assessing LLM Responses

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Measures the correctness of the advice provided, ensuring it is factually correct and free from errors.</td>
</tr>
<tr>
<td>Relevance</td>
<td>Assesses how well the advice addresses the given prompt, ensuring that the response is contextually appropriate and pertinent to the security issue at hand.</td>
</tr>
<tr>
<td>Comprehensiveness</td>
<td>Evaluates the extent to which the advice covers all necessary aspects of the security issue, ensuring no critical information is omitted.</td>
</tr>
<tr>
<td>Adherence to Best Practices</td>
<td>Ensures the advice aligns with established secure coding standards, such as the OWASP Secure Coding Practices, providing practical and applicable guidance.</td>
</tr>
</tbody>
</table>

Visualizations such as bar charts and scatter plots were used to illustrate the comparative performance of the models across various evaluation criteria. By employing rigorous statistical methods, the study aimed to provide a detailed and nuanced understanding of the strengths and weaknesses of each LLM in providing secure software development advice. The benchmarking process not only highlighted the areas where each model excelled but also identified opportunities for further improvement.

1) Data Collection:
   a) Collect responses from ChatGPT and Google Gemini for each prompt.
   b) Organize responses systematically for subsequent analysis.

2) Preprocessing:
   a) Clean and normalize the collected responses to ensure consistency.
   b) Tokenize and encode the responses as necessary for statistical analysis.

3) Evaluation Metrics Calculation:
   a) Compute the accuracy score for each response.
   b) Assess the relevance of each response by matching it to the prompt context.
   c) Evaluate the comprehensiveness by ensuring all critical aspects of the security issue are addressed.
   d) Measure adherence to best practices by comparing responses to established secure coding guidelines.

4) Statistical Analysis:
   a) Perform t-tests to compare the mean scores of ChatGPT and Google Gemini across all metrics.
   b) Conduct ANOVA to analyze the variance within and between the models’ performance on different metrics.
   c) Apply post-hoc tests to further investigate significant differences identified by ANOVA.

5) Visualization:
   a) Generate bar charts to depict the average scores of each model for all evaluation metrics.
   b) Create scatter plots to illustrate the relationship between different metrics for each model.
   c) Use box plots to present the distribution of scores, highlighting variations and outliers.

6) Interpretation and Reporting:
   a) Summarize the statistical findings, identifying the strengths and weaknesses of each model.
   b) Discuss the practical implications of the results for secure software development.
   c) Suggest areas for improvement and future research based on the observed performance patterns.

By following these structured steps, the study ensures a rigorous and systematic approach to comparing and benchmarking the performance of ChatGPT and Google Gemini. The use of advanced statistical techniques and comprehensive visualizations provides a robust framework for understanding the models’ capabilities and limitations in delivering secure software development advice. This detailed methodology highlights the thoroughness of the evaluation process and the reliability of the findings.

IV. RESULTS

The results section presents the benchmarking outcomes of ChatGPT and Google Gemini, focusing on the defined evaluation metrics. The performance of each model was assessed across various criteria, and the findings are detailed below with accompanying tables and figures to illustrate the comparative performance.

A. Performance Metrics

The performance metrics included accuracy, relevance, comprehensiveness, and adherence to best practices. The table below summarizes the average scores for each model across these metrics.
To provide a more detailed visualization of the performance differences, the following figure depicts the average scores for each metric using bar charts.

The results indicated that Google Gemini generally outperformed ChatGPT across all evaluation metrics. The most significant difference was observed in the accuracy metric, where Google Gemini achieved a score of 88.1%, compared to ChatGPT’s 85.4%. Similarly, for relevance, Google Gemini scored 85.3%, slightly higher than ChatGPT’s 82.7%. In terms of comprehensiveness, Google Gemini also led with 80.5% compared to ChatGPT’s 78.9%. Finally, for adherence to best practices, Google Gemini achieved 84.2%, whereas ChatGPT scored 81.6%.

### B. Strengths

The strengths of each model were identified through a detailed analysis of their performance across the various metrics. ChatGPT demonstrated strong performance in generating contextually relevant responses, which is critical for addressing specific security prompts accurately. Its ability to provide comprehensive advice covering multiple aspects of security issues was also notable, particularly in scenarios requiring detailed explanations and step-by-step guidance.

Google Gemini excelled in accuracy and adherence to best practices, ensuring the advice given was not only correct but also aligned with established secure coding standards. This capability is crucial for developers seeking reliable guidance that adheres to industry norms. Additionally, Google Gemini’s higher scores in relevance and comprehensiveness indicated its proficiency in providing well-rounded and thorough responses, making it a valuable resource for comprehensive security advice.

The table above provides a detailed statistical analysis of the performance metrics for ChatGPT and Google Gemini. The mean scores and standard deviations (SD) for each metric are presented, along with the p-values and effect sizes for the comparative analysis. The p-values indicate the statistical significance of the differences observed, while the effect sizes measure the magnitude of these differences.

ChatGPT’s strong performance in generating contextually relevant responses, as indicated by a relevance score of 82.7 ± 3.1, highlights its ability to address specific security prompts accurately. Its comprehensiveness score of 78.9 ± 3.7 further shows its capability to provide detailed and thorough advice, particularly useful in scenarios requiring step-by-step guidance and in-depth explanations. Google Gemini, on the other hand, demonstrated superior performance in accuracy and adherence to best practices, with scores of 88.1 ± 2.1 and 84.2 ± 2.6, respectively. These metrics are crucial for developers seeking reliable and standards-aligned guidance. The higher relevance score of 85.3 ± 2.8 and comprehensiveness score of 80.5 ± 3.2 indicate Google Gemini’s proficiency in providing well-rounded and thorough responses, making it a valuable resource for comprehensive security advice. The response time metric further adds to the evaluation, with Google Gemini exhibiting a slightly faster response time (240 ± 12 ms) compared to ChatGPT (250 ± 15 ms). While both models demonstrated efficient response times, the slightly faster performance of Google Gemini may enhance its usability in real-time advisory scenarios.

### C. Weaknesses

The weaknesses of each model were also scrutinized to identify areas for improvement. ChatGPT, while proficient in relevance and comprehensiveness, showed a slight lag in accuracy compared to Google Gemini. This gap suggests that there may be instances where the advice generated by ChatGPT, though contextually appropriate, may not always be factually correct, necessitating further verification by developers.

Google Gemini, despite its strengths, exhibited a minor deficiency in providing highly contextual responses, particularly in complex scenarios requiring nuanced understanding. Although its adherence to best practices was commendable, the slight dip in comprehensiveness compared to its other metrics highlighted areas where additional detail and depth could enhance the overall utility of its advice.

The table above provides a comprehensive analysis of the weaknesses of ChatGPT and Google Gemini, including detailed performance metrics, ranges, and variance. The mean scores and standard deviations (SD) highlight the areas where each model exhibited deficiencies. For instance, ChatGPT’s accuracy, with a mean of 85.4 ± 2.5, was lower than Google Gemini’s 88.1 ± 2.1, indicating potential issues with factual correctness that might require developers to verify the advice independently.

The contextual understanding metric further shows the challenges faced by both models, with ChatGPT scoring 77.2 ± 4.0 and Google Gemini 79.0 ± 3.5. This metric, crucial for complex scenarios requiring nuanced responses, showed significant variability, as evidenced by the high variance values. The ranges for each metric also provide insight into the consistency of the models’ performance, with Google Gemini generally exhibiting tighter ranges and thus more consistent results. While Google Gemini’s adherence to best practices was strong, its slight deficiency in comprehensiveness (80.5 ± 3.2) compared to its accuracy (88.1 ± 2.1) suggests that while it adheres well to established standards, it may lack depth in some responses. This gap highlights the need for further enhancement in providing detailed and exhaustive advice, ensuring that all critical aspects of security issues are addressed thoroughly. The response time metric, although secondary to the content quality, is also significant for practical applications. ChatGPT’s mean response time was slightly higher at 250 ± 15 ms compared to Google Gemini’s 240 ± 12 ms, with a broader range indicating less consistency. Although both models demonstrated efficient response times, this slight difference may impact real-time usability in advisory scenarios.
Fig. 1. Comparative Performance of ChatGPT and Google Gemini Across Metrics

### TABLE IV
PERFORMANCE METRICS AND STATISTICAL ANALYSIS OF CHATGPT AND GOOGLE GEMINI

<table>
<thead>
<tr>
<th>Metric</th>
<th>ChatGPT (Mean ± SD)</th>
<th>Google Gemini (Mean ± SD)</th>
<th>p-value</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>85.4 ± 2.5</td>
<td>88.1 ± 2.1</td>
<td>0.002</td>
<td>0.6</td>
</tr>
<tr>
<td>Relevance</td>
<td>82.7 ± 3.1</td>
<td>85.3 ± 2.8</td>
<td>0.005</td>
<td>0.5</td>
</tr>
<tr>
<td>Comprehensiveness</td>
<td>78.9 ± 3.7</td>
<td>80.5 ± 3.2</td>
<td>0.03</td>
<td>0.3</td>
</tr>
<tr>
<td>Adherence to Best Practices</td>
<td>81.6 ± 2.9</td>
<td>84.2 ± 2.6</td>
<td>0.004</td>
<td>0.55</td>
</tr>
<tr>
<td>Response Time (ms)</td>
<td>250 ± 15</td>
<td>240 ± 12</td>
<td>0.01</td>
<td>0.4</td>
</tr>
</tbody>
</table>

### TABLE V
DETAILED PERFORMANCE METRICS ANALYSIS INCLUDING RANGES AND VARIANCE

<table>
<thead>
<tr>
<th>Metric</th>
<th>ChatGPT (Mean ± SD)</th>
<th>Google Gemini (Mean ± SD)</th>
<th>Range (ChatGPT)</th>
<th>Range (Gemini)</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>85.4 ± 2.5</td>
<td>88.1 ± 2.1</td>
<td>82.0 - 89.0</td>
<td>85.5 - 91.0</td>
<td>6.25</td>
</tr>
<tr>
<td>Relevance</td>
<td>82.7 ± 3.1</td>
<td>85.3 ± 2.8</td>
<td>79.0 - 87.5</td>
<td>82.0 - 89.0</td>
<td>9.61</td>
</tr>
<tr>
<td>Comprehensiveness</td>
<td>78.9 ± 3.7</td>
<td>80.5 ± 3.2</td>
<td>74.0 - 84.0</td>
<td>76.0 - 85.0</td>
<td>13.69</td>
</tr>
<tr>
<td>Adherence to Best Practices</td>
<td>81.6 ± 2.9</td>
<td>84.2 ± 2.6</td>
<td>78.0 - 85.0</td>
<td>81.0 - 88.0</td>
<td>8.41</td>
</tr>
<tr>
<td>Response Time (ms)</td>
<td>250 ± 15</td>
<td>240 ± 12</td>
<td>230 – 270</td>
<td>220 – 255</td>
<td>225</td>
</tr>
<tr>
<td>Contextual Understanding</td>
<td>77.2 ± 4.0</td>
<td>79.0 ± 3.5</td>
<td>72.0 – 82.0</td>
<td>74.0 – 84.0</td>
<td>16.0</td>
</tr>
</tbody>
</table>
V. Discussion

The benchmarking results provide valuable insights into the capabilities and limitations of ChatGPT and Google Gemini in delivering secure software development advice. The comparative analysis has highlighted distinct strengths and areas for improvement for each model, contributing to a deeper understanding of their potential roles as advisory tools. The superior performance of Google Gemini in accuracy and adherence to best practices shows its utility for developers seeking reliable and standards-compliant guidance. In contrast, ChatGPT’s notable performance in generating contextually relevant and comprehensive responses makes it a valuable resource for addressing complex security issues that require detailed explanations.

The implications of these findings for software developers are multifaceted. Developers can leverage the strengths of Google Gemini to obtain precise and standards-aligned advice, which is crucial for ensuring the security and robustness of their software systems. The high accuracy and adherence to best practices demonstrated by Google Gemini suggest that developers can rely on it for guidance that conforms to industry norms, thereby reducing the likelihood of introducing security vulnerabilities. On the other hand, ChatGPT’s ability to provide comprehensive and contextually relevant responses makes it an excellent tool for scenarios where detailed understanding and multi-faceted advice are necessary. Developers can benefit from using ChatGPT to gain deeper insights into complex security issues and to explore various aspects of secure coding practices.

The study’s findings also have significant implications for the future development and improvement of large language models in the domain of secure software development. Enhancing the accuracy of ChatGPT while maintaining its strengths in relevance and comprehensiveness could lead to a more balanced and versatile advisory tool. Similarly, improving the contextual understanding and depth of responses generated by Google Gemini would enhance its utility in addressing intricate security issues. Future research could focus on refining the algorithms and training data used for these models to better capture the nuances of secure software development and to provide more precise and contextually appropriate advice.

In addition to model improvements, the study suggests several avenues for future research in benchmarking large language models. One potential area of exploration is the development of more sophisticated evaluation metrics that capture the complexity and multi-dimensionality of secure software development advice. Metrics that assess the long-term impact of the advice on software security, as well as its applicability in real-world scenarios, would provide a more holistic assessment of the models’ capabilities. Furthermore, expanding the scope of benchmarks to include a wider range of secure coding standards and guidelines from various domains could enhance the comprehensiveness of the evaluation.

The methodology employed in this study could also be refined and expanded in future research. Incorporating additional automated analysis techniques, such as machine learning-based evaluation methods, could provide more nuanced insights into the quality and effectiveness of the LLM responses. Additionally, exploring the use of real-time feedback mechanisms, where the models are continuously updated based on developer feedback, could improve the relevance and applicability of the advice provided. By integrating these enhancements, future studies could offer a more robust and dynamic evaluation of large language models in the context of secure software development.

Finally, the practical recommendations based on the study results emphasize the importance of integrating large language models into the software development lifecycle as supplementary tools. Developers should utilize the strengths of both ChatGPT and Google Gemini to address different aspects of secure software development, thereby enhancing the overall security posture of their projects. Organizations should consider incorporating these models into their development workflows, alongside traditional security practices, to leverage their potential for providing timely and relevant advice. By doing so, developers can achieve a more comprehensive approach to security, ensuring that their software systems are resilient against emerging threats.

VI. Conclusion

The study has provided a comprehensive evaluation of ChatGPT and Google Gemini in the domain of secure software development advice, highlighting their respective strengths and weaknesses across various performance metrics. Through rigorous benchmarking, it was observed that Google Gemini excelled in accuracy and adherence to best practices, making it a highly reliable source of standards-compliant guidance for developers. This capability is particularly crucial for ensuring that the advice given aligns with established secure coding norms, thereby mitigating the risk of introducing security vulnerabilities. The higher scores in relevance and comprehensiveness further reinforce Google Gemini’s proficiency in delivering well-rounded and thorough responses, essential for comprehensive security advice. Conversely, ChatGPT demonstrated a strong performance in generating contextually relevant and comprehensive responses, which is vital for addressing specific security prompts with detailed explanations. Its ability to provide multi-faceted advice covering various aspects of security issues indicates its utility in scenarios requiring in-depth understanding and step-by-step guidance. Despite its lower accuracy compared to Google Gemini, ChatGPT’s relevance and comprehensiveness scores show its potential as a valuable tool for developers seeking detailed and contextually appropriate advice. The statistical analysis and visualizations presented in the results section have elucidated the distinct advantages of each model, providing a nuanced understanding of their capabilities and limitations. The inclusion of metrics such as accuracy, relevance, comprehensiveness, and adherence to best practices has allowed for a multidimensional assessment of the models’ performance, ensuring that the evaluation captures not only the factual correctness but also the practical applicability and completeness of the advice. The detailed analysis of response times further adds to the
The creation of resilient and robust software systems that can withstand the ever-evolving landscape of cybersecurity threats.

The implications of the study’s findings for the role of large language models in secure software development are significant. By leveraging the strengths of both ChatGPT and Google Gemini, developers can enhance their coding practices and ensure adherence to security standards. The ability of Google Gemini to provide accurate and standards-aligned advice makes it an indispensable resource for developers seeking reliable guidance, while ChatGPT’s strength in generating comprehensive and contextually relevant responses offers valuable insights for addressing complex security issues. The complementary nature of the models’ capabilities suggests that a combined approach, utilizing both models’ strengths, can provide a robust framework for secure software development. The benchmarking study has showed the potential of large language models as advisory tools in the field of secure software development. The detailed evaluation and comparative analysis have provided a thorough understanding of the models’ performance, informing their future application and development. By harnessing the capabilities of ChatGPT and Google Gemini, developers can achieve a more comprehensive and secure approach to software development, ensuring the creation of resilient and robust software systems that can withstand the ever-evolving landscape of cybersecurity threats.

REFERENCES