

The Current State of Generative Artificial Intelligence Tools for Accessibility in Product Development

Iyad Abu Doush

College of Engineering and Applied Sciences

American University of Kuwait

idoush@auk.edu.kw

Salmiya, Kuwait

Abstract- This paper addresses the pressing need to evaluate the maturity and performance metrics of generative AI tools dedicated to accessibility in product development. The problem lies in the lack of standardized methods for assessing the maturity of generative AI tools tailored to accessibility needs and the absence of universally accepted performance metrics to measure their efficacy. This deficiency hampers the advancement of inclusive design practices and limits the potential impact of AI-driven accessibility solutions. This paper proposes a comprehensive framework for evaluating the maturity of AI tools specifically designed for accessibility in product development. We elucidate the critical criteria integral to this evaluation, encompassing aspects such as usability, reliability, scalability, and adaptability to diverse user needs and contexts. The proposed solution aims to contribute valuable knowledge to the evolving landscape of generative AI tools dedicated to enhancing accessibility in product development. By establishing a structured approach to assessing maturity and advocating for standardized performance metrics, our research endeavors to empower developers, designers, and stakeholders to make informed decisions regarding the adoption and refinement of AI-driven accessibility solutions.

Keywords- ChatGPT; Copilot; Digital accessibility; AI; Code generation; Accessible product development.

1. Introduction

Numerous research endeavors have focused on enhancing developer productivity through various approaches such as code synthesis, code search, and other forms of code recommender systems [1]. Many of these initiatives harness the capabilities of Artificial Intelligence, particularly employing deep learning techniques [2]. A significant development in this domain occurred in June 2021 when GitHub and OpenAI introduced GitHub Copilot—a revolutionary "AI pair programmer" compatible with several IDEs. Fueled by the expansive OpenAI Codex model, trained on a vast corpus of open-source GitHub code. Copilot excels in proposing code snippets across various programming languages¹. Additionally, ChatGPT, another powerful tool in this realm, offers a

¹ <https://docs.github.com/en/copilot/overview-of-github-copilot/about-github-copilot-individual>

conversational interface that can be leveraged for code generation and product development [3]. These tools leverage vast datasets to create generative models, with the aim of producing code that facilitates the development of accessible products. However, the maturity of these tools remains a critical consideration.

Copilot provides users with three primary functionalities: converting comments to code, suggesting tests that match implementation code, and auto-filling for repetitive code. The conversion of comments to code, which is initiated when a user writes a comment describing the logic they intend to implement. Although a Copilot suggestion can be triggered by a natural language comment alone, optimal results are achieved when users supplement their input with meaningful names for function parameters and descriptive comments [2].

ChatGPT can be used to generate code by engaging in a conversational interaction with the model. The user needs to clearly express the intent for code generation. Then furnish relevant context and details about the code to generate. Finally, interact with ChatGPT through a series of prompts and responses to refine the code [3].

The use of Large Language Models (LLM), like ChatGPT, to automatically improve web accessibility explored in the work [4]. The paper compares ChatGPT's effectiveness in detecting and fixing accessibility issues to manual testing on two non-compliant websites. Results show promise for ChatGPT in enhancing web accessibility, vital for meeting WCAG 2.1 guidelines and creating more inclusive online platforms. The burgeoning field of Artificial Intelligence (AI) has witnessed a substantial integration of tools aimed at fostering accessibility in product development.

In light of this inquiry, this research addresses several key questions. Firstly, it explores effective methods for assessing the maturity of generative AI tools dedicated to accessibility in product development and elucidates the criteria integral to this evaluation. Secondly, the paper delves into the diverse performance metrics employed to gauge the effectiveness and efficiency of these tools and proposes potential standardization measures. This exploration aims to contribute valuable knowledge to the evolving landscape of generative AI tools dedicated to enhancing accessibility in product development.

Various metrics serve as benchmarks to evaluate the effectiveness and inclusivity of generative AI tools' code outcomes. Measures encompassing code functionality, visual structure, complexity, consideration of multiple input devices, and the severity level of accessibility issues within the generated code are pivotal for comprehensive assessment. By incorporating these multifaceted measures into the evaluation process, stakeholders can make informed decisions regarding the selection and optimization of AI tools for accessibility in product development. This holistic approach ensures the delivery of inclusive digital experiences that cater to the diverse needs of users across different abilities.

2. Generative AI to Improve Code Accessibility

In recent years, the integration of artificial intelligence (AI) within software development has heralded significant advancements in accessibility. Among these innovations, generative AI stands out as a transformative force, particularly in the realm of code accessibility. This section explores the burgeoning landscape of Generative AI and its profound implications for enhancing accessibility in product development. By automating the creation and modification of code structures, Generative AI not only expedites the development process but also fosters inclusivity by empowering developers of all skill levels to create accessible software. Within this context, we delve into four distinct cases where Generative AI demonstrates its efficacy [5].

- A. **Code Generation for Accessibility:** AI tools utilize extensive datasets to generate code that fosters accessibility in product development. However, the maturity of these tools is still evolving. One key challenge lies in their ability to accurately describe content through code. While tools can suggest accessibility attributes, integrating them with precise values based on content remains a hurdle. The following is an example use case of a prompt to generate accessible code using Copilot:

Prompt: *Act as accessibility coach subject matter expert to create accessible HTML table that conform with WCAG 2.1 A and AA success criteria. The table has 5 rows and 3 columns with the headers Name, Email, and Date of birth. When possible, use semantic HTML to ensure it is operable using the keyboard. Also, follow ARIA Authoring Practices guide and related design patterns.*

Generated Code: Here's an example of an accessible table with the specified headers and content:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Accessible Table Example</title>
</head>
<body>
  <table role="grid" aria-label="User Information">
    <thead>
      <tr>
        <th scope="col">Name</th>
        <th scope="col">Email</th>
        <th scope="col">Date of Birth</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>John Doe</td>
```

The Current State of Generative Artificial Intelligence Tools for Accessibility in Product Development

```
        <td><a
href="mailto:john@example.com">john@example.com</a></td>
        <td>1985-07-15</td>
    </tr>
    <tr>
        <td>Jane Smith</td>
        <td><a
href="mailto:jane@example.com">jane@example.com</a></td>
        <td>1990-03-22</td>
    </tr>
    <!--Add more rows as needed -->
</tbody>
</table>
</body>
</html>
```

- B. Accessibility Feedback:** When evaluating accessibility issues within code, the reliability of feedback from AI tools is still a concern. For instance, these tools may struggle to determine if the values assigned to attributes accurately describe the content. This uncertainty poses challenges for developers relying on AI-generated feedback to ensure accessibility compliance.

Prompt: *Act as accessibility coach subject matter expert to identify accessibility issues and how to fix them in the following HTML table. This is to ensure that the code conform with WCAG 2.1 A and AA success criteria. When possible, check the use of semantic HTML to ensure it is operable using the keyboard. Also, ensure it follows ARIA Authoring Practices guide and related design patterns. The following is the HTML table code:*

```
<table>
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
  <tr>
    <td>Row 3, Column 1</td>
    <td>Row 3, Column 2</td>
  </tr>
</table>
```

Response: The following is the Recommendations section of the response:

- Add a <caption> element to describe the purpose of the table (e.g., “Sales Data for Q1 2024”).
- If the table has headers, use <th> elements for them.

- Provide alternative text for any non-text content (if applicable).
- Test keyboard navigation within the table.

C. **Repair Recommendations and Guidance:** An essential aspect of AI tools in product development is their capacity to provide repair recommendations. While tools can identify accessibility issues, the ability to guide developers on fixing these issues is crucial.

D. **Creation of Test Cases:** AI tools are expected not only to identify accessibility issues but also to actively contribute to the creation of accessible code and test cases. The maturity of these tools is reflected in their ability to not only point out problems but also propose solutions.

One of the main challenges of utilizing AI tools for software development of accessible products is to integrate content and code generation in a way that ensures semantic and structural coherence. For example, an AI tool may be able to suggest accessibility attributes for HTML elements, such as alt, aria-label, or role, but it may not be able to assign appropriate values to these attributes based on the content and context of the web page. Moreover, an AI tool may not be able to verify if the values provided by the developer are accurate and descriptive enough for assistive technologies to help users with disabilities. Therefore, an AI tool should not only generate code, but also provide feedback and repair recommendations to help developers fix accessibility issues and improve code quality.

3. Performance Indicators for Code Accessibility

Assessing the efficacy of AI tools in ensuring code accessibility presents a multifaceted challenge. Currently, there exists a lack of standardized methodologies for evaluating code accessibility, necessitating the identification of suitable performance indicators. These indicators encompass various aspects, such as the quantity and severity of accessibility errors detected by the tool, the comprehensiveness and precision of the generated code, the satisfaction and usability of individuals with disabilities interacting with the product, and the adherence to pertinent accessibility standards and guidelines. However, it's imperative to recognize that these indicators can be influenced by factors like the nature, purpose, and complexity of the product, as well as the characteristics of its target audience and user scenarios. Thus, an effective AI tool must demonstrate adaptability across diverse contexts and deliver actionable insights to both developers and stakeholders.

In evaluating the performance of AI tools for enhancing accessibility in product development, a holistic approach is necessary. This approach involves considering various metrics to gauge the tool's effectiveness. Metrics may include precision in identifying accessibility issues, accuracy in suggesting relevant accessibility attributes, and the integration of context-based values to ensure the accuracy of content. Additionally, the tool's effectiveness in providing actionable recommendations for repairs, its capability to

generate accessible code and test cases, and its ability to provide insightful feedback on the overall accessibility level of the code are crucial factors to assess. By comprehensively evaluating these aspects, stakeholders can gain a deeper understanding of the tool's impact on enhancing code accessibility and ensure that it meets the diverse needs of users across different contexts and scenarios.

Previous work in the area of measuring AI generated code includes [6] presenting measure for evaluating quality of generated code to be compilation and runtime errors, wrong outputs, code style, maintainability, and efficiency. Another work [7] proposes to use validity, correctness, efficiency in terms of time and space complexity.

In this work we propose a set of measures to evaluate the accessibility of the generated code. These measures are not presented previously in the literature. Firstly, the visual order of the code plays a crucial role in enhancing readability and comprehension, particularly for developers and individuals utilizing screen readers. Tools should prioritize generating code with a clear and logical structure, facilitating easier navigation and comprehension. Secondly, the consideration of multiple input devices broadens the accessibility scope, accommodating users with varying abilities. AI tools should generate code that supports diverse input modalities, including keyboard navigation, voice commands, and gesture controls, fostering an inclusive user experience. Lastly, evaluating the severity level of accessibility issues within the generated code aids in prioritizing and addressing critical barriers to access. The code generated by the tools needs to be evaluated in terms of the severity of accessibility shortcomings, empowering developers to allocate resources effectively and prioritize remediation efforts.

4. Conclusion and Future Work

In conclusion, while generative AI tools for accessibility in product development exhibit considerable promise, their current stage of development raises concerns regarding their reliability and effectiveness. Overcoming technical and methodological challenges, such as integrating content and code generation, providing robust feedback and repair recommendations, creating suitable test cases and metrics, and adapting to diverse contexts and requirements, remains paramount for their advancement. Achieving this goal necessitates further research and collaboration among researchers, developers, users, and accessibility experts.

Moving forward, it is imperative to focus on enhancing the reliability of feedback mechanisms, refining repair recommendations, and ensuring the accurate integration of attribute values within AI tools for accessibility. The proposed performance indicators can serve as a valuable benchmark for evaluating and advancing the maturity of these tools in facilitating accessibility in product development. Furthermore, the proposed standardization measures require validation through expert review and real-world case studies to ensure their practical relevance and applicability. One promising avenue for advancing the field of Artificial Intelligence Tools for Accessibility in Product

Development is the establishment of a comprehensive "golden dataset" comprised of accessible code samples. This dataset would serve as a benchmark for evaluating the efficacy of generative AI tools in producing accessible code and identifying areas for improvement. Additionally, testing the ability of current generative AI tools to generate code that meets accessibility standards and comparing the results against this golden dataset presents an exciting opportunity for future research.

Performance measures outlined in the present study can be utilized to quantify any shortcomings observed in the code generated by AI tools. By systematically analyzing the generated code against the golden dataset, researchers can identify patterns of deficiencies and assess the extent to which current generative AI tools fall short in producing accessible code. This analysis can provide valuable insights into areas requiring further development and refinement.

Through ongoing research and collaboration, we can drive forward the evolution of AI tools for accessibility, ultimately fostering a more inclusive and equitable digital landscape for all users.

References

- [1] S. Luan, D. Yang, C. Barnaby, K. Sen, and S. Chandra, 'Aroma: Code recommendation via structural code search', *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, pp. 1–28, 2019.
- [2] N. Nguyen and S. Nadi, 'An empirical evaluation of GitHub copilot's code suggestions', in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 1–5.
- [3] N. M. S. Surameery and M. Y. Shakor, 'Use chat gpt to solve programming bugs', *Int. J. Inf. Technol. Comput. Eng.*, no. 31, pp. 17–22, 2023.
- [4] A. Othman, A. Dhouib, and A. Nasser Al Jabor, 'Fostering websites accessibility: A case study on the use of the Large Language Models ChatGPT for automatic remediation', in *Proceedings of the 16th International Conference on Pervasive Technologies Related to Assistive Environments*, 2023, pp. 707–713.
- [5] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, 'Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation', *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.
- [6] Y. Liu et al., 'Refining ChatGPT-generated code: Characterizing and mitigating code quality issues', *ACM Trans. Softw. Eng. Methodol.*, 2023.
- [7] B. Yetistiren, I. Ozsoy, and E. Tuzun, 'Assessing the quality of GitHub copilot's code generation', in *Proceedings of the 18th international conference on predictive models and data analytics in software engineering*, 2022, pp. 62–71.