



PDF Download
3748699.3749805.pdf
19 December 2025
Total Citations: 0
Total Downloads: 54



Published: 03 September 2025

[Citation in BibTeX format](#)

GoodIT '25: International Conference on Information Technology for Social Good
September 3 - 5, 2025
Antwerp, Belgium

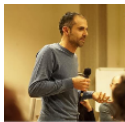
Conference Sponsors:
SIGCAS

Latest updates: <https://dl.acm.org/doi/10.1145/3748699.3749805>

RESEARCH-ARTICLE

An Assessment of LLM-Based Auditing and Validation for Web Accessibility

BARRY BASSI, University of Bologna, Bologna, BO, Italy



LLM for Web Accessibility

GIOVANNI DELNEVO, University of Bologna, Bologna, BO, Italy

MIRKO FRANCO, University of Padua, Padua, PD, Italy

OMBRETTA GAGGI, University of Padua, Padua, PD, Italy

SALVATORE GATTO, University of Padua, Padua, PD, Italy

SILVIA MIRRI, University of Bologna, Bologna, BO, Italy

[View all](#)

Open Access Support provided by:

[University of Bologna](#)

[University of Padua](#)

An Assessment of LLM-Based Auditing and Validation for Web Accessibility

Barry Bassi
University of Bologna
Bologna, Italy
barry.bassi@unibo.it

Ombretta Gaggi
University of Padua
Padua, Italy
gaggi@math.unipd.it

Giovanni Delnevo
University of Bologna
Bologna, Italy
giovanni.delnevo2@unibo.it

Salvatore Gatto
University of Padua
Padua, Italy
salvatore.gatto@math.unipd.it

Mirko Franco
University of Padua
Padua, Italy
mifranco@math.unipd.it

Silvia Mirri
University of Bologna
Bologna, Italy
silvia.mirri@unibo.it

Kelvin Olaiya
University of Bologna
Bologna, Italy
kelvin.olaiya@unibo.it

Abstract

Accessibility constitutes a fundamental right codified in national and international legislation and the SDGs. In particular, web accessibility is essential to enable people with disabilities to engage fully in societal and professional activities. Despite mandatory legal requirements, such as those included in the Web Accessibility Directive and the European Accessibility Act, a substantial portion of websites continue to exhibit accessibility barriers. Moreover, awareness of accessibility best practices remains limited among designers and developers. In this context, we assess the effectiveness of Large Language Models for automated accessibility auditing and HTML validation. We also examine how these models can aid developers in producing semantically correct and accessible web content, and we discuss their current limitations. Our research highlights the promises and challenges of integrating LLMs in web development and advocates using these tools as supportive aids rather than standalone solutions.

CCS Concepts

• **Human-centered computing** → **Accessibility**: *Accessibility design and evaluation methods*; • **Computing methodologies** → **Natural language generation**.

Keywords

web accessibility, large language models, HTML, WCAG, digital sustainability

ACM Reference Format:

Barry Bassi, Giovanni Delnevo, Mirko Franco, Ombretta Gaggi, Salvatore Gatto, Silvia Mirri, and Kelvin Olaiya. 2025. An Assessment of LLM-Based Auditing and Validation for Web Accessibility. In *International Conference*

on Information Technology for Social Good (GoodIT '25), September 03–05, 2025, Antwerp, Belgium. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3748699.3749805>

1 Introduction

Web accessibility is a fundamental right which guarantees equal access to digital services and resources for all individuals, including those with disabilities. This right is particularly crucial for people who depend on remote access to essential services due to mobility limitations or other barriers [23]. As society has become increasingly digital over the past few decades, the need for accessible digital environments has grown significantly. Numerous institutions and organizations have advocated this cause. The World Wide Web Consortium (W3C), for instance, developed the Web Content Accessibility Guidelines (WCAG) to provide a global standard for accessible web design [13]. Organizations like WebAIM (Web Accessibility in Mind) further support this mission by offering tools, training, and resources to promote inclusive design practices [16]. Despite these efforts, accessibility remains a widespread issue. According to WebAIM's annual report analyzing the top one million websites, the 96.8% still fail to meet basic accessibility standards. This reality is in clear opposition to the principles outlined in the United Nations Convention on the Rights of Persons with Disabilities, which asserts the right to accessible digital content [23]. In response, significant legislative progress has been made—particularly in Europe [2, 8, 20, 22]. The European Disability Strategy 2010–2020 identified web accessibility as one of its most successfully achieved objectives [21]. This was largely due to the adoption of the Web Accessibility Directive, which established a robust legal and technical framework mandating that all public sector websites and mobile applications comply with accessibility standards [8]. This initiative was further strengthened by the European Accessibility Act, which extended these requirements to key private sector services—such as transportation and banking—and applied them to companies with annual revenues exceeding 500 million euros [20]. Together, these legislative measures mark a crucial step toward a more inclusive digital society.



This work is licensed under a Creative Commons Attribution 4.0 International License. *GoodIT '25, Antwerp, Belgium*

© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2089-5/25/09
<https://doi.org/10.1145/3748699.3749805>

However, a major challenge persists: the lack of awareness and expertise among web designers and developers. Many are unfamiliar with international guidelines and national regulations and often do not consider accessibility as an integral part of their design and development process. Even when they are aware, there is frequently a lack of understanding of the basic tests that websites must pass, as well as a shortage of effective tools that can automatically detect and highlight critical accessibility issues.

Gaggi and Pederiva [11] defined 150 tests to cover all the accessibility issues and showed that a complete and automatic tool does not exist. A test can be classified as *manual*, *semi-automatic* and *automatic*. Most tests (56%) need human interaction because they evaluate if the information on a page are understandable or ask the developer to perform a specific task, e.g., verifying the possibility of surfing the website only using a keyboard. 40% of the tests can be classified as *semi-automatic*, i.e., a tool can identify a specific problem. Still, human interaction is needed to evaluate the output of the test: e.g., most tools report when the same link anchor is used more than once on a web page and people using screen readers can be disoriented when the same anchor brings them to different destinations. More advanced tools can report which titles are used for repeated anchors, but only a human being can establish if an alternative title is efficient and understandable for users. Moreover, web developers must be able to understand when tools report false positives, e.g., an image without an alternative text is correct if it is used only for decoration purposes.

Only a small subset of the tests (4%) is completed *automatically*, such as the ones that validate the page to the standard or control the presence of broken links. Automated tools reduce the time needed to perform a web accessibility evaluation because they are much faster than human revision, and therefore provide a great help to developers but do not completely solve the problem of accessibility testing. According to [11] the most complete tool is the Arc Toolkit developed by The Paciello Group, which covers only 26% of the tests. Since no tool covers all the needed tests, webmasters need to know a lot of tools, to integrate their features and eventually manage different outputs on the same issue. Moreover, many validation tools give output that is not easy to understand and must be interpreted. This requires a lot of effort, time, knowledge and skills. In this paper, we explore the potential of Large Language Models (LLMs) to enhance web accessibility testing and support the design process. Our primary objective is to assess their effectiveness in identifying compliance with the Web Content Accessibility Guidelines (WCAG) and their ability to provide clear, informative explanations of accessibility issues for both developers and designers. Our evaluation followed a two-step approach. First, we examined the capability of LLMs to validate short HTML code snippets that commonly lead to accessibility problems—such as missing labels for input elements or improper nesting of HTML tags. In the second phase, we extended our analysis to more complex, real-world scenarios by evaluating websites developed as final projects in a Web Technologies course at University of Padua. Finally, we analyzed the validation reports generated by the models, focusing on the clarity and usefulness of their explanations. The results demonstrate that LLMs are capable of accurately interpreting the semantics of HTML elements and assessing their compliance with WCAG 2.1 and supporting error correction by providing potential solutions to identified problems.

This work helps to achieve SDG #10 “Reduced Inequality”, #11 “Sustainable Cities and Communities” and #16 “Peace and Justice Strong Institutions” and #17 “Strengthening global partnerships for sustainable development by contributing to inclusive solutions”.

The remainder of this paper is organized as follows. Section 2 presents a review of the relevant literature. Section 3 describes the methodological details of this work. The results are reported and discussed in Section 4. Finally, we draw our conclusions and present some future research directions in Section 5.

2 Related Work

The recent enactment of accessibility policies and regulatory frameworks, such as the Web Accessibility Directive (WAD) and the European Accessibility Act (EAA), has significantly increased the need to expedite the development of products and services - including websites - that adhere to accessibility standards. In this context, researchers have made considerable efforts to develop solutions that support developers in making the web more accessible for all. For example, Gaggi *et al.* [11, 12] introduced *WCAG4All*, a new tool designed to help web developers in understanding and complying with web accessibility guidelines. *WCAG4All* provides a comprehensive set of tests to ensure compliance with WCAG 2.1 and Italian regulations, offering a user-friendly approach. This tool is not fully automated and requires human oversight, particularly for more complex accessibility assessments, such as evaluating semantics and verifying color contrast.

The rapid diffusion and adoption of Large Language Models (LLMs) has catalyzed the development of novel solutions across numerous domains, including (but not limited to) healthcare [10, 25], content moderation [9, 14], education [24], and coding and web accessibility [1, 4, 6, 17, 19]. Prominent examples of LLMs include the GPT series developed by OpenAI (e.g., GPT-4.1, o4-mini, etc.) and Meta’s LLaMa (Large Language Model Meta AI) family.

Large language models can be used to assist in automating certain web validation tasks and promoting adherence to the Web Content Accessibility Guidelines (WCAG), potentially enhancing the efficiency (and effectiveness) of accessibility evaluations. For example, Delnevo *et al.* [4] investigated the capabilities of ChatGPT in detecting and addressing web accessibility issues. The study presented a range of both successful and unsuccessful cases, highlighting the model’s ability to identify common accessibility violations such as missing alternative text and improper semantic structures within HTML code. While the findings indicated promising potential, the model exhibited notable limitations when processing more complex HTML structures, resulting in various inaccuracies and errors. Based on these observations, the authors concluded that although ChatGPT is not suitable for fully replacing manual code review, it can nonetheless serve as a valuable assistive tool in the accessibility evaluation process. Similarly, Bassi *et al.* [1] evaluated the capabilities of LLMs, namely LLaMa 3.1 and GPT-4o, in detecting errors in simple HTML snippets and in generating accessible HTML code for basic elements, such as tables, forms, and images. Instead, Duarte *et al.* [6] conducted a test campaign involving 10 web pages to evaluate the capabilities of LLMs, specifically LLaMa 3.1, GPT-4o, and GPT-4o mini, in detecting issues related to web page headings. While the authors recognize the potential of LLMs in

accessibility evaluation, they also acknowledge the need for human intervention and emphasize the importance of human-machine collaboration. In this direction, Mowar *et al.* [17] presented *CodeA11y*, an innovative GitHub Copilot Extension designed to assist developers in creating accessible websites by suggesting accessible code and providing reminders for manual verification. Acknowledging the limited awareness of accessibility, especially among novice developers, the authors conducted a user study demonstrating that developers are more likely to produce accessible code when using *CodeA11y*.

Drawing on the findings of the aforementioned prior studies, our research aims to further evaluate the capabilities of various LLMs in identifying errors in HTML code and conducting accessibility assessments in accordance with the WCAGs. In particular, we focus on models developed by Meta and OpenAI, given their widespread adoption, to better understand the current state of these tools and their potential in automating accessibility validation.

3 Methodology

In this section, we introduce the dataset used for our preliminary evaluation, the considered Large Language Models (LLMs) and the prompt employed in the experiments.

3.1 Datasets

To assess the capabilities of the considered LLMs in the HTML validation and accessibility audit, we constructed a dataset composed of (1) snippets of not-valid HTML code, which include common errors made by developers which add a failure in the compliance with WCAGs (e.g., tags without the matching end tag, missing legend in the fieldset tag, etc.), and (2) simple web sites developed as final projects of the Web Technologies course of the Degree in Computer Science at the University of Padua.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Simple HTML Example</title>
6 </head>
7 <body>
8   <h1>This is a title</h1>
9   <h2>This is another title</h2>
10  <h3>This is a subtitle</h3>
11  <p>This is a simple text paragraph</p>
12  <p>Another paragraph with <strong>important</strong> and <em>emphasized</em>
13  parts</p>
14  <p>This is a sentence written in <span lang="it">italiano but this other
15  words are not.</p>
16 </html>

```

Listing 1: Example of HTML snippet

Listing 1 reports an example of HTML code: we can note that some tags (e.g., tag `<h2>` line 9, tag `<p>` line 11 and tag `` line 13) do not have a matching end tag (only self-containing tag, such as `` and `<input>` can avoid the end tag). Despite the simplicity of this example, the correctness of headings tags is essential for supporting the navigation of people with disabilities, considering that screenreaders use headings to navigate webpages successfully. Moreover, screenreaders are not able to switch again to English language on line 13 since the end tag for the `span` is missing.

The dataset contains seven examples designed to showcase different HTML elements, including tables, images, ordered and unordered lists, headings, and forms. This approach ensured a comprehensive evaluation of various HTML structures. Moreover, it also contains three complete websites. To give an idea of the complexity of these websites, they have respectively 32, 31, and 21 different pages, containing up to 600 lines of code each. These tests allowed us to test LLMs on both practical and controlled scenarios thereby offering a more robust analysis of their performance across different levels of HTML complexity. Figure 1 shows an example of the home page of a website included in our dataset.

3.2 Large Language Models employed

We employed LLaMa 3.1 (405B) [7], GPT-3.5 Turbo [3], and GPT-4o [18] - significant size models trained over large-scale *corpora* - to perform a preliminary evaluation of their capabilities in understanding and validating HTML code, even considering accessibility guidelines, as well as in generating correct and accessible code. GPT-4o is one of OpenAI's flagship models that can reason across any combination of text, audio, images, and videos, showing advanced understanding capabilities. GPT-3.5 is a less powerful model, accepting text as input. Instead, LLaMa 3.1 is an open alternative developed by Meta and one of the most adopted open foundation models. We believe that the research community can benefit from our choice of including an open model in our evaluation, considering the increasing need for transparency in data management.

3.3 Methods and Prompts

The prompt we used to analyze the snippets of HTML code is:

User: Validate the following HTML code and provide a summary of changes. The code is: `html_code`

where `{html_code}` is the HTML code to evaluate. Once tested the models with simple examples, we moved to more complex websites. In this case, we also considered the CSS and JavaScript files, thus improving the accuracy of the analysis, e.g., for text-background contrast issues and the functionality of control elements. Due to the large amount of contextual code, the prompts consist of a relatively high number of tokens, close to the maximum allowed by the models.

Accessibility validation is carried out in two steps to detect as many violations as possible and to identify hallucinations. In the first step, a prompt is used to obtain the WCAG 2.1 AA guideline violations detected on the individual web pages. The prompt is:

Step 1: Validation of an HTML page: You are a tool for detailed web accessibility validation. Your task is to analyze the provided HTML, CSS, and JavaScript code and return a report listing all accessibility issues based on the WCAG 2.1 guidelines at the AA level. I ask you to generate a summary table in Markdown format, structured as follows:
 {TABLE_DATA_DESCRIPTION}
 {HTML_PAGE}
 {EXTERNAL_CSS_AND_JS}

In the second step, a more specific prompt is used for each detected violation, to verify whether the reported issue is real or is a

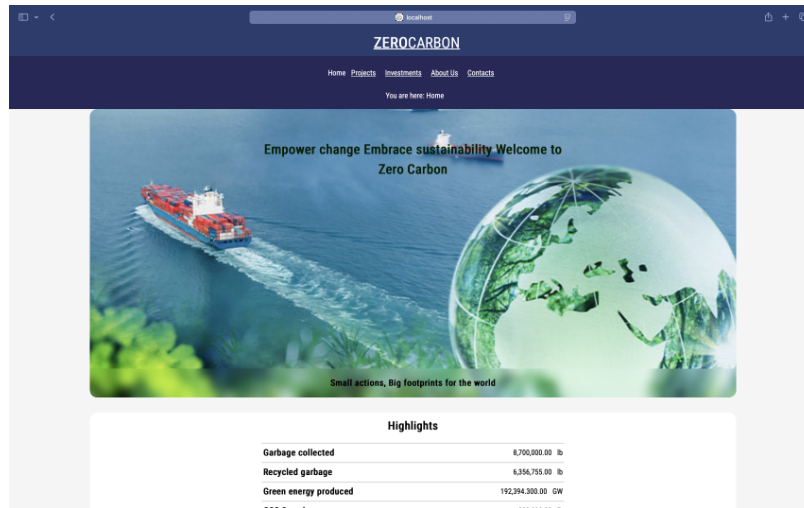


Figure 1: Homepage of a website included in our dataset

hallucination. The idea is to follow a sort of Chain-of-Verification approach [5]. It is a method where LLM must check the facts it provides step by step. By verifying each part of its response, the LLM can reduce mistakes and improve the accuracy of its answers.

This second step consists of several stages:

- (1) we manually classified each issue as correct or hallucination;
- (2) we analyzed individually each identified violation of step 1 to let the LLM classify it as correct or hallucination;
- (3) we evaluated the accuracy of the LLM in identifying hallucinations by contrasting its results against the labels assigned during the first stage;
- (4) finally, an evaluation of the performance of the LLMs at step 1 is carried out. The prompt is:

Step 2: Verification of an accessibility issue detected in Step 1:

Given this accessibility issue according to WCAG 2.1 and the page where it was detected (including external CSS and JavaScript files), could you explain in more detail what it refers to and whether it is a real issue or not?

```
{SINGLE_ISSUE_FROM_GPT_MODEL}
{HTML_PAGE}
{EXTERNAL_CSS_AND_JS}
```

Given the ability of the LLMs to provide a more efficient explanation for developers of the detected violation, and considering the need to teach developers correct practices to write accessible and valid code, we performed a last test to assess if LLMs can help in generating HTML code, using some common use cases that usually create accessibility issues, such as forms and tables. To this aim, we asked LLMs to answer the following prompt:

User: Explain to me how to write an accessible [form, paragraph, table] in HTML, which is compliant with the WCAG 2.1, including an example

4 Results and Discussion

In this section, we report the results of our evaluation of the capabilities of LLMs in validating HTML code and accessibility auditing, and we discuss how these models are a valuable tool for supporting the development of correct and accessible websites.

4.1 HTML code snippets

The results conducted for this research showed positive results, despite some imprecision. The initial experiments consisted of evaluating Meta’s model Llama 3.1 and OpenAI’s model GPT-4o for HTML validation. Correct HTML validation is crucial as it facilitates the identification of potential accessibility issues in web content. Both models proved their ability to properly validate HTML snippets by identifying common errors, such as missing closing tags, invalid HTML structures and improper nesting, and provided a list of necessary changes to correct the code.

However, as the complexity of the HTML snippets increased, their accuracy and precision in error detection decreased. Both models had distinct strengths and weaknesses. Llama 3.1 was more focused on the identification of high-level issues, often providing recommendations related to accessibility, e.g., the addition of `aria-label` attributes, or suggesting best practices for attributes. The model tended to overlook more intricate syntax or structural issues and it often failed to detect specific errors, e.g., missing closing tags or mismatched elements. This behaviour highlighted the need to ask further questions to get a proper answer: in Listing 2 Llama 3.1 needed our help to detect the unclosed `<p>`. But, the HTML code provided as a solution is correct.

```
1 <footer>
2   <div class="author">
3     <p>By <strong>Amanda</strong>
4   </div>
5   <a href="" target="_blank" aria-label="Read more about the article (opens in a
  new window)" class="refLink">Read more...</a>
6 </footer>
```

Listing 2: The `<p>` tag is not closed in this snippet.

GPT-4o generally performs better than Llama 3.1 in complex scenarios, identifying a wider range of errors. It is more accurate in the detection of common issues like mismatched or missing tags, especially when HTML snippets involve deeply nested elements or more complex structures. However, GPT-4o did exhibit occasional hallucinations—reporting errors that were not present in the code. For example, in Listing 3 GPT-4o highlighted the error “*The <h3> tag in the Garbage collected article ends with an <h4>. Change it to <h3>.*” This error, however, was a hallucination, since line 4 does not contain an <h4>. OpenAI’s model wrongly referred to the error present in line 16, where a tag <h3> is closed by </h4>. This mistake highlights some limitations in the model’s ability to localize errors in complex scenarios. Despite these occasional hallucinations, GPT-4o provided more detailed feedback than Llama 3.1. It effectively identified issues such as improperly closed tags, inconsistent attribute values, missing quotation marks, and naming inconsistencies. Its granular feedback makes it more reliable overall.

```

1 <section id="highlights" class="h1Bg">
2   <h2>Highlights</h2>
3   <article class="highlight">
4     <h3>Garbage collected</h3>
5     <p>8,700,000.00 <abbr title="Pounds">lb</abbr></p>
6   </article>
7   <article class="highlight">
8     <h3>Click here to go the next page!</h3>
9     <p>6,356,755.00 <abbr title="Pounds">lb</abbr>
10  </p>
11  </article>
12  <article class="highlight">
13    <h3>Green energy produced</h3>
14    <p>192,394.300.00 <abbr title="Gigawatt">GW</abbr></p>
15  </article>
16  <article class="highlight">
17    <h3>CO2 Saved</h4>
18    <p>992,080.00 <abbr title="Pounds">lb</abbr>
19    </p>
20  </article>
21  <p id="lastUp">Last updated: <time datetime="2024-01-12">12-January-2024</time>
22 </p>
23 </section>

```

Listing 3: Example of GPT-4o’s hallucination.

With this premise, we decide to further explore OpenAI’s models effectiveness in analyzing complex websites, incorporating the Web Content Accessibility Guidelines (WCAG) into the evaluation.

4.2 Websites

We validated all the HTML pages of the three websites included in our dataset. The results of the first step are reported in Table 1. Overall, GPT-4o detected more violations than GPT-3.5 (i.e., 429 violations against 182 ones). It is interesting to notice that there is just one violation relative to guideline 2.4.9 that was detected by GPT-3.5 and not by GPT-4o. All the other ones were detected by GPT-4o. Many of them, for instance, 14 out of 28, were just detected by GPT-4o and not by GPT-3.5. The experiments seem to suggest the ability of both models to interpret the semantics of HTML elements on a page concerning the WCAG 2.1 guidelines. They seem capable of analyzing labels and ARIA attributes in relation to the page context, as well as identifying text-background contrast issues and significant structural errors in the code.

We then moved on to the second step of our approach. After having manually classified all violations raised by both GPT-3.5 and GPT-4o, we let GPT-4o classify as correct violations or hallucinations, all the violations of the previous step, taking advantage of the prompt presented in Subsection 3.3. We decided to just use

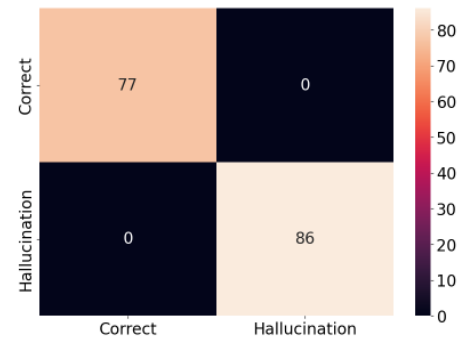


Figure 2: Confusion matrix of GPT-4o used to detect hallucinations of GPT3.5

GPT-4o for this critical task, since it is a more accurate model than the 3.5 version, being a bigger LLM. It is important to note that in a relatively small number of cases (19 for GPT3.5 and 44 for GPT-4o), GPT-4o raised some doubts, being unable to assess the detected violation. Hence, we marked these violations as “Need Manual Verification” and excluded them from the next step. The rationale behind this choice derives from the fact that we believe that LLMs should be support tools for developers. Hence, being able to ask for human collaboration in the assessment problem is considered acceptable.

Once each violation had been classified, we contrasted the prediction of GPT-4o against our ones on both violations identified by GPT-3.5 and GPT-4o. The corresponding confusion matrixes are reported respectively in Figures 2 and 3. As shown, there is complete agreement between the human assessment and that provided in the second step by GPT-4o, resulting in a 100% of accuracy. This may be due to several reasons. First of all, the Chain-of-Verification technique [5] has already proven to be an effective prompt engineering technique. Moreover, a prompt detailing a specific task with a limited length could have also influenced the final results. On the other hand, such an impressive result is however limited to a small set of projects. A more in-depth evaluation has to be carried out in order to be able to generalize this finding.

In the end, thanks to the previous stages, it has been possible to assess the performance of GPT-3.5 and GPT-4o during the first step. Thanks to the human and GPT-4o evaluations, it was possible to determine both the number of correct violations detected and the hallucinations. Such results are reported in Table 2, where each project reports the number of correct violations detected and hallucinations for each model together with the “Need Manual Verification” cases. As shown, in the first step, the two models perform significantly differently. GPT-3.5 got a number of hallucinations that were close to 50% while GPT-4o hallucinated just in 15% of violations. It is also interesting to notice that for the evaluations provided by both models, GPT-4o label violations as “Need Manual Verification” in about 10% of the cases.

Such experiments demonstrated that LLMs could be successfully employed for detecting accessibility issues in large HTML websites. The proposed two-step pipeline, based on the Chain-of-Verification approach [5], can guarantee, in the end, that almost all the raised

Table 1: WCAG 2 violations with frequencies per model

Violation (SC)	Description	GPT-3.5	GPT-4o
1.1.1 Non-text Content	Text alternatives for non-text content	9	114
1.3.1 Info and Relationships	Programmatically determinable info/relationships	54	42
1.3.5 Identify Input Purpose	Input purposes programmatically determined	0	2
1.4.1 Use of Color	Color not sole means of info	0	1
1.4.3 Contrast (Minimum)	Sufficient contrast for text/images	26	48
1.4.4 Resize Text	Text resizable without assistive tech	1	7
1.4.5 Images of Text	Avoid using images of text	0	1
1.4.10 Reflow	Content reflows when resized	0	2
1.4.12 Text Spacing	No loss when text spacing adjusted	0	1
1.4.13 Content on Hover or Focus	Hover/focus content is dismissible	0	2
2.1.1 Keyboard	All functionality via keyboard	0	7
2.1.2 No Keyboard Trap	No keyboard traps	0	2
2.4.1 Bypass Blocks	Skip repetitive content blocks	0	28
2.4.2 Page Titled	Pages have descriptive titles	0	1
2.4.3 Focus Order	Focus order preserves meaning	5	2
2.4.4 Link Purpose (In Context)	Link purpose clear from context	35	54
2.4.6 Headings and Labels	Headings/labels describe topic	3	4
2.4.7 Focus Visible	Focus indicator is visible	2	19
2.4.9 Link Purpose (Link Only)	Link purpose from link text alone	6	0
2.5.3 Label in Name	Accessible name matches label	0	2
3.1.2 Language of Parts	Language of parts determined	0	3
3.2.2 On Input	Input change doesn't alter context	1	2
3.2.3 Consistent Navigation	Navigation order consistent	1	0
3.3.1 Error Identification	Input errors identified	0	8
3.3.2 Labels or Instructions	Labels/instructions for input	8	6
4.1.1 Parsing	Content parses without errors	0	8
4.1.2 Name, Role, Value	UI components' name/role/value determined	30	61
4.1.3 Status Messages	Status messages programmatically determined	1	2
Total		182	429

Table 2: Verification results for detected issues across all projects

Model	Result	Project 1		Project 2		Project 3		Overall	
		No.	%	No.	%	No.	%	Total	%
GPT-3.5	Correct	18	45.00%	42	56.76%	17	25.00%	77	42.31%
	Hallucination	21	52.50%	26	35.14%	39	57.35%	86	47.25%
	Needs Manual Verification	1	2.50%	6	8.11%	12	17.65%	19	10.44%
	Total	40	100%	74	100%	68	100%	182	100%
GPT-4o	Correct	66	75.86%	130	77.38%	122	70.11%	318	74.13%
	Hallucination	19	21.84%	15	8.93%	33	18.97%	67	15.62%
	Needs Manual Verification	2	2.30%	23	13.69%	19	10.92%	44	10.25%
	Total	87	100%	168	100%	174	100%	429	100%

issues are correct and not due to hallucinations. Even if GPT-4o achieved better results than GPT-3.5 during the first step, the use of the latter should be considered. In fact, not only it is a cheaper model, but also it consumes less energy, being a more sustainable choice. Even if it produces more hallucinations, they can be addressed in the second step, using GPT-4o.

The use of LLMs in this context brings several advantages. They can be easily adapted and updated to follow the latest WCAG guidelines. Additionally, they can support error correction by providing potential solutions to identified problems, can analyze the content

of multiple files simultaneously and provide unified reports, offer customization of reports according to specific needs, and can be integrated into the development process, providing real-time feedback on accessibility during code writing.

4.3 HTML Generation

The third experiment consists of testing Llama 3.1 and GPT-4o with the generation of HTML snippets which implements key web features, such as tables, forms, images, ordered and unordered lists.

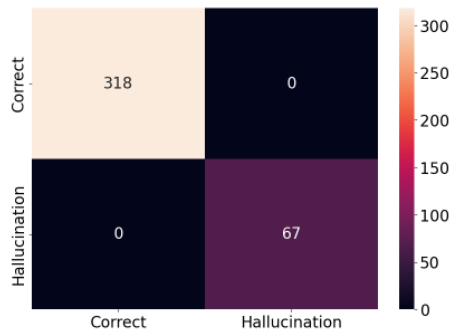


Figure 3: Confusion matrix of GPT-4o used to detect hallucinations of GPT-4o

The aim was to assess the ability of each model to produce HTML code that conforms to WCAG 2.1, focusing on compliance with accessibility standards and code accuracy. In particular, we asked both models the following questions:

- write an accessible table in HTML compliant with WCAG 2.1;
- write a form in HTML compliant with WCAG 2.1;
- write a list of images in HTML compliant with WCAG 2.1.

This last experiment showed a higher level of compliance in the HTML snippets code generated by GPT4o- compared to Llama3.1. For instance, let’s analyzed the answers of both model to the generation of a form compliance with WCAG2.1. Listing 4 reports the code generated by Llama 3.1: the form may appear functional, but it doesn’t respect some important accessibility guidelines: it lacks the <fieldset> and <legend> elements, which are crucial for users relying on screen readers. These elements group related fields together and provide descriptive context, making it easier for users with disabilities to understand the form’s structure. Therefore, Llama 3.1’s output does not fully meet the WCAG 2.1 standards.

```

1 <form>
2 <h2>Contact Us</h2>
3 <label for="name">Name:</label>
4 <input type="text" id="name" name="name" aria-label="Your name">
5 <br>
6 <label for="email">Email:</label>
7 <input type="email" id="email" name="email" aria-label="Your email address">
8 <br>
9 <label for="message">Message:</label>
10 <textarea id="message" name="message" aria-label="Your message"></textarea>
11 <br>
12 <button type="submit">Send</button>
13 </form>
    
```

Listing 4: Example of form generated by Llama 3.1.

Listing 5 reports the code generated by GPT-4o. The form includes <fieldset> and <legend> which group related fields together and provide clear context, making the form more understandable and navigable for users with disabilities, and compliant with WCAG 2.1.

```

1 <form action="/submit" method="POST">
2 <fieldset>
3 <legend>Personal Information</legend>
4 <label for="fname">First Name:</label>
5 <input type="text" id="fname" name="fname" autocomplete="given-name"
6   required>
7 <label for="lname">Last Name:</label>
8 <input type="text" id="lname" name="lname" autocomplete="family-name"
9   required>
    
```

```

8 <label for="email">Email:</label>
9 <input type="email" id="email" name="email" autocomplete="email" required>
10 </fieldset>
11 <button type="submit">Submit</button>
12 </form>
    
```

Listing 5: Example of form generated by GPT-4o.

However, GPT-4o’s solution also doesn’t contain any aria-label attributes, but it just relies on the use of <label> tags. Llama’s form instead used unnecessary aria-label attributes, which makes redundant the reading of such elements by screen-reader. It is widely recognized that users with visual impairments often require more time to complete tasks due to the need for assistive technologies and alternative methods of accessing information. Therefore, the use of redundant tags has a negative impact on efficiency, increasing the time required to complete a task.

Lastly, GPT-4’s form included the required and autocomplete attributes, which follows the best practice for web accessibility. In contrast, Meta’s solution lacks of these features.

In conclusion, the results demonstrated that OpenAI’s GPT-4o has an advanced ability to generate accessible and precise HTML code snippets, which reinforces its position as a superior model for tasks requiring strict compliance with accessibility guidelines.

4.4 Limitations

Despite advanced (language) understanding capabilities exhibited by these models, several issues and limitations remain. For example, LLMs fall short in generating content that require domain-specific knowledge, as in this specific context. In addition, these models sometimes provide the correct answer following an invalid reasoning path, leading to inconsistencies and contradictions between the final answer and the reasoning process. Moreover, LLMs often generate untruthful information that is not coherent with existing knowledge (i.e., hallucinations). These models also suffer from knowledge recency, i.e., they have difficulties managing recent knowledge (after the cut-off date), as discussed in [26]. One possible solution is the use of Retrieval Augmented Generation (RAG) [15], i.e., including some additional context into the prompt, retrieving information from a base of knowledge created starting from some external and relevant documents.

The main limitation of this work regards the dataset employed for the evaluation. It contains a limited amount of code snippets and websites. Moreover, snippets have been created ad-hoc while websites are University projects and not real websites. To better generalize our findings, more code snippets and websites should be included in the analysis. This could be beneficial for several reasons. On the one hand, since these types of models do not produce deterministic output, they can generate responses with hallucinations. We took advantage of a prompt engineering technique to alleviate this problem but more tests are needed to confirm the positive results that emerged during our experiments. On the other hand, real websites could not be analysed due to the constraints on the number of tokens the models can process.

Another limitation regards a lack of comparison with traditional validators. Understanding the extent to which LLMs could improve the validation process with respect to traditional approaches has to be established. Moreover, a combination of traditional approaches

and a validation based on LLMs could be envisioned, to improve the quality of the process, minimizing the environmental impact.

5 Conclusion

Accessibility is a fundamental right for all citizens, recognized by national and international regulations, such as the European Accessibility Act, whose aim is to remove the barriers created by divergent rules of EU members, as well as to increase the number of accessible products and services in the market. Unfortunately, accessibility by law is not sufficient. Indeed, both technical and non-technical people, including developers and designers, lack knowledge of accessibility and its principles.

In this context, we have discussed the use of LLMs to support the development of correct and accessible websites, assessing the capabilities of some selected language models in finding issues in HTML code and generating examples of appropriate and accessible HTML snippets to support developers during their work and teach them correct web development practices. Indeed, despite their advanced language understanding capabilities, we cannot fully rely on LLMs to validate HTML code and perform accessibility auditing. However, as we have demonstrated in this work, these models can be a valuable support in spreading accessibility culture among developers and designers.

We plan to extend our work in several directions. First of all, we plan to increase the number of examples in our database and perform a more in-depth analysis of LLMs, as well as to consider other LLMs (e.g., Google's Gemma, Mistral AI, etc.). To overcome some of the well-known issues of LLMs, we would also like to use the aforementioned Retrieval Augmented Generation (RAG), retrieving information from some relevant documents, such as the HTML standard, the WCAGs, and other material created specifically for this purpose. Moreover, we plan to consider an even more recent (draft) version of the WCAGs. Finally, we aim to assess the capabilities of LLMs in the discussed task across diverse languages.

Acknowledgments

This research was partially supported by the Department of Mathematics of the University of Padua (SID project: “A tool to spread accessibility culture through gamification”). Responsibility of the content resides with the authors.

References

- [1] Barry Bassi, Giovanni Delnevo, Mirko Franco, Ombretta Gaggi, Salvatore Gatto, Silvia Mirri, and Kelvin Olaiya. 2025. Supporting Accessibility Auditing and HTML Validation using Large Language Models. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing* (Catania International Airport, Catania, Italy) (SAC '25). Association for Computing Machinery, New York, NY, USA, 27–31. doi:10.1145/3672608.3707912
- [2] US Congress. 1998. Section 508 of the Rehabilitation Act. <https://www.law.cornell.edu/uscode/text/29/794d>
- [3] Context.ai. 2024. GPT-3.5 Turbo System Card. <https://context.ai/model/gpt-3-5-turbo/>.
- [4] Giovanni Delnevo, Manuel Andruccioli, and Silvia Mirri. 2024. On the Interaction with Large Language Models for Web Accessibility: Implications and Challenges. In *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*. 1–6. doi:10.1109/CCNC51664.2024.10454680
- [5] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason E Weston. [n. d.]. Chain-of-Verification Reduces Hallucination in Large Language Models. In *ICLR 2024 Workshop on Reliable and Responsible Foundation Models*.
- [6] Carlos Duarte, Miguel Costa, Leticia Seixas Pereira, and João Guerreiro. 2025. Expanding Automated Accessibility Evaluations: Leveraging Large Language Models for Heading-Related Barriers. In *Companion Proceedings of the 30th International Conference on Intelligent User Interfaces (IUI '25 Companion)*. Association for Computing Machinery, New York, NY, USA, 39–42. doi:10.1145/3708557.3716329
- [7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and Ahmad Al-Dahle et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] <https://arxiv.org/abs/2407.21783>
- [8] European Parliament. 2016. Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies. <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016L2102>
- [9] Mirko Franco, Ombretta Gaggi, and Claudio E. Palazzi. 2024. Integrating Content Moderation Systems with Large Language Models. *ACM Trans. Web* (Oct. 2024). doi:10.1145/3700789 Just Accepted.
- [10] Marco Furini, Michele Mariani, Sara Montagna, and Stefano Ferretti. 2024. Conversational Skills of LLM-based Healthcare Chatbot for Personalized Communications. In *Proceedings of the 2024 International Conference on Information Technology for Social Good* (Bremen, Germany) (GoodIT '24). Association for Computing Machinery, New York, NY, USA, 429–432. doi:10.1145/3677525.3678693
- [11] Ombretta Gaggi and Veronica Pederiva. 2021. WCAG4All, a tool for making web accessibility rules accessible. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. 1–6. doi:10.1109/CCNC49032.2021.9369484
- [12] Ombretta Gaggi and Lorenzo Perinello. 2022. Improving accessibility of web accessibility rules. In *Proceedings of the 2022 ACM Conference on Information Technology for Social Good* (Limassol, Cyprus) (GoodIT '22). Association for Computing Machinery, New York, NY, USA, 167–174. doi:10.1145/3524458.3547267
- [13] World Wide Web Consortium Web Accessibility Initiative Group. 2023. Web Content Accessibility Guidelines (WCAG) 2.2. <https://www.w3.org/TR/WCAG22/>
- [14] Mahi Kolla, Siddharth Salunkhe, Eshwar Chandrasekharan, and Koustuv Saha. 2024. LLM-Mod: Can Large Language Models Assist Content Moderation?. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI EA '24). Association for Computing Machinery, New York, NY, USA, Article 217, 8 pages. doi:10.1145/3613905.3650828
- [15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.
- [16] WebAIM Million. 2022. The 2022 report on the accessibility of the top 1,000,000 home pages. <https://webaim.org/projects/million/>
- [17] Peya Mowar, Yi-Hao Peng, Jason Wu, Aaron Steinfeld, and Jeffrey P Bigham. 2025. CodeA11y: Making AI Coding Assistants Useful for Accessible Web Development. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 45, 15 pages. doi:10.1145/3706598.3713335
- [18] OpenAI. 2024. GPT-4o System Card. <https://openai.com/index/gpt-4o-system-card/>.
- [19] Achraf Othman, Amira Dhouib, and Aljazi Nasser Al Jabor. 2023. 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* (Corfu, Greece) (PETRA '23). Association for Computing Machinery, New York, NY, USA, 707–713. doi:10.1145/3594806.3596542
- [20] European Parliament. 2019. Directive (EU) 2019/882 of the European Parliament and of the Council of 17 April 2019 on the accessibility requirements for products and services. <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32019L0882>
- [21] European Parliament. November. Commission Staff Working Document, “Evaluation of the European Disability Strategy 2010-2020s”. <https://ec.europa.eu/social/main.jsp?catId=89&furtherNews=yes&newsId=9835&langId=en>
- [22] Italian Parliament. 2004. Legge 9 gennaio 2004, n. 4. https://www.gazzettaufficiale.it/atto/serie_generale/caricaDettaglioAtto/originario?atto.dataPubblicazioneGazzetta=2004-01-17&atto.codiceRedazionale=004G0015&elenco30giorni=false
- [23] United Nations. 2006. Convention on the Rights of Persons with Disabilities.
- [24] Xinyu Jessica Wang, Christine P. Lee, and Bilge Mutlu. 2025. LearnMate: Enhancing Online Education with LLM-Powered Personalized Learning Plans and Support. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (CHI EA '25). Association for Computing Machinery, New York, NY, USA, Article 373, 10 pages. doi:10.1145/3706599.3719857
- [25] Xuhai Xu, Bingsheng Yao, Yuanzhe Dong, Saadia Gabriel, Hong Yu, James Hendler, Marzyeh Ghassemi, Anind K. Dey, and Dakuo Wang. 2024. Mental-LLM: Leveraging Large Language Models for Mental Health Prediction via Online Text Data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 1, Article 31 (March 2024), 32 pages. doi:10.1145/3643540

[26] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang,

Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL] <https://arxiv.org/abs/2303.18223>