# Supporting Accessibility Auditing and HTML Validation using Large Language Models

Barry Bassi
Dept. Computer Science Engineering
University of Bologna
Bologna, Italy
barry.bassi@unibo.it

Giovanni Delnevo
Dept. Computer Science Engineering
University of Bologna
Bologna, Italy
giovanni.delnevo2@unibo.it

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

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

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

Silvia Mirri
Dept. Computer Science Engineering
University of Bologna
Bologna, Italy
silvia.mirri@unibo.it

Kelvin Olaiya
Dept. Computer Science Engineering
University of Bologna
Bologna, Italy
kelvin.olaiya@unibo.it

## Abstract

Accessibility is a fundamental right for all citizens, as recognized by national and international regulations. In particular, web accessibility is crucial for people with disabilities to participate fully in society and the workforce. Unfortunately, imposing accessibility by law is insufficient, considering the number of websites still presenting relevant accessibility issues. On the other hand, there is a lack of culture about accessibility, even among designers and developers. In this context, we evaluate the capabilities of Large Language Models (LLMs) in accessibility auditing and HTML validation. We also discuss how these tools can support developers in building correct and accessible websites and their limitations.

## 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


*SAC'25, March 31 –April 4, 2025, Sicily, Italy*

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/xx.xxx/xxx_x

**ACM Reference Format:**
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 ACM SAC Conference (SAC'25).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/xx.xxx/xxx_x

## 1 Introduction

Web accessibility plays a crucial role in recognizing the rights of people with disabilities [25] since they most need access to information without moving from their homes. Web accessibility also represents a complex challenge: on the one hand, legislation, especially in Europe, defines clear and strong rules and expands its application area [4, 9, 22, 24]; on the other, many studies still report that a large number of websites are still not accessible [1–3, 19, 27].

The European Disability Strategy 2010-2020 [23] states that web accessibility was one of the most successfully achieved goals in terms of legislation. This result was due to the approval of the Web Accessibility Directive [9], which set a technical and legal framework, and imposed the accessibility requirements to all the EU public administration websites and mobile apps, and to the following European Accessibility Act [22], which expanded the minimum accessibility requirements to other subjects, like transport or banking, and to all the companies with annual revenue greater than 500 million euros.

But imposing accessibility by law is not enough: according to the last report on the top one million pages by WebAIM [17], the number of pages that had a failure during Web Content Accessibility Guidelines (WCAG) [14] tests stood at 96.8%, which is a little better than the previous year but still an enormous number.

The main problem is the lack of knowledge about accessibility among web designers and developers: they are often not aware of either international guidelines about accessibility or national

regulations. For this reason, they are often unable to consider accessibility during the development and to compile a complete list of tests that the website must pass. Therefore, they cannot find accessibility errors on their websites. Moreover, a tool that automatically tests all accessibility issues does not exist and developers often do not know which tool can be used to test each feature. Gaggi and Pederiva [12] 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 [12] 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 aim at lowering the effort required to test a website and at improving accessibility culture among web developers using capabilities of *Large Language Models* (*LLMs)* to provide a suitable output that non-experts can understand. The goal is not only to test the conformance with the WCAGs but also to use these tools, in case of errors, to explain them to web developers. For this reason, we start by validating short HTML pieces of code. Even if this first test can be done with an HTML validator, many accessibility issues arise from HTML errors, e.g., the absence of a label for an input tag, or the incorrect nesting of HTML tags. Moreover, we want to explore the capabilities of the LLMs to provide a validation report which can explain the error to the developer, so we first need to find accessibility issues. Then our second test considered more complex input. Therefore, we analyzed the website developed as the final project for a class on Web Technologies at [institution's name removed for anonymization].

Our results show the strong ability of Large Language Models (LLMs) to interpret the semantics of HTML elements on a page concerning the WCAG 2.1 guidelines. Additionally, they support 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

Upon the recent introduction of accessibility initiatives and regulations, such as the European Union's Accessibility Act, accelerating the development of websites that are compliant with accessibility guidelines has become increasingly important. In this context, researchers have proposed several solutions to support developers in making the web more accessible for all. For example, Gaggi *et al.* [12, 13] introduced *WCAG4All*, a new tool designed to assist 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. The tool was developed before the release of LLMs and is not fully automated. It requires human oversight, particularly for more complex accessibility evaluations, such as those related to semantics and color contrast verification.

The widespread adoption of Large Language Models (LLMs) has then enabled previously unimaginable scenarios; nowadays, these models are being applied in several areas, such as healthcare [11, 26], content moderation in online social networks [10, 18], coding and web accessibility [6, 21], etc. Some LLMs are OpenAI's GPT series (e.g., GPT 3.5, GPT-4o, etc.) and Meta's LlaMa collection.

These systems may be employed as tools for automating web validation tasks and ensuring compliance with the Web Content Accessibility Guidelines (WCAG). For example, Delnevo *et al.* [6] conducted a study on the ability of ChatGPT to evaluate and correct web accessibility issues. This research presented a series of successful and unsuccessful cases, demonstrating ChatGPT's capacity to identify missing alt text, incorrect semantic structures in HTML code, and other accessibility violations. Although the results were considered promising, OpenAI's model encountered significant challenges when analyzing more complex HTML code, leading to various inaccuracies and errors. As results, the authors concluded that while ChatGPT cannot fully replace manual code review but it can serve as a valuable support tool.

The release of new LLMs has made this research field highly dynamic, with significant advancements occurring within just a few months. For instance, López-Gil *et al.* [16] tested GPT-4's ability to detect compliance with specific WCAG success criteria, sharing the same objective as [6] in automating the process of WCAG conformity verification. Their study employed a more recent and advanced OpenAI's model, namely GPT-4, and it is focused on the compliance with a smaller sample of WCAG success criteria: 1.1.1 (Non-text Content), 2.4.4 (Link Purpose), and 3.1.2 (Language of Parts). Their findings indicate that GPT-4 can successfully detect accessibility issues often missed by current automated tools, achieving an 87.18% detection accuracy across test cases.

Considering the findings of these studies, our research aims to valuate the ability of different LLMs to validate web pages compliance with WCAGs. We decided to focus on Meta's and OpenAI's models for their widespread diffusion. We aim to understand the current state of performance of these tools and their potential for automating accessibility validation. Through this research, we hope to contribute valuable insights into the strengths and limitations of LLMs in the context of web accessibility.

## 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 (10) 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 [institution's name removed for anonymization]. The source code of the examples included in our dataset is available at [link removed for anonymization]. As already discussed, this is not a mere HTML validation but our examples include HTML errors that raise accessibility issues.

```
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>emphatized</em>
           parts</p>
13     <p>This is a sentence written in <span lang="it">italiano but this other
         words are not.</p>
14 </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 <img> 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) [8], GPT-3.5 Turbo [5], and GPT-4o [20] - 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 hallucination. The idea is to follow a sort of Chain-of-Verification approach [7]. 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;

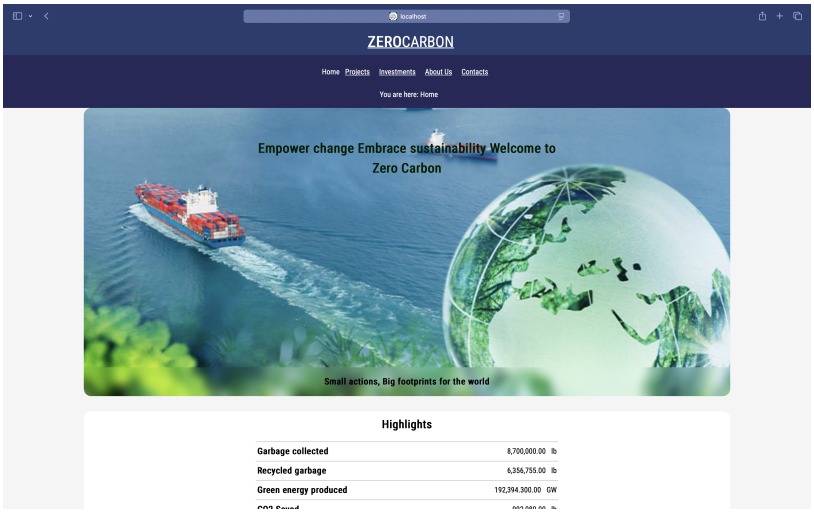


**Figure 1: Homepage of a website included in our dataset**

(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 reffered 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.

```
<section id="highlights" class="hlBg">
  <h2>Highlights</h2>
    <article class="highlight">
      <h3>Garbage collected</h3>
      <p>8,700,000.00 <abbr title="Pounds">lb</abbr></p>
    <article>
    <article class="highlight">
      <h3>Click here to go the next page!</h3>
      <p> 6,356,755.00 <abbr title="Pounds">lb</abbr>
    </article>
    <article class="highlight">
      <h3>Green energy produced</h3>
      <p>192,394.300.00 <abbr title="Gigawatt">GW</abbr></p>
    </article>
    <article class="highlight">
      <h3>CO2 Saved</h4>
        <p>992,080.00 <abbr title="Pounds">lb</abbr>
              </p>
    </article>
      <p id="lastUp">Last updated: <time datetime="2024-01-12">12-January-2024>
        </p>
</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 the three websites 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 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.

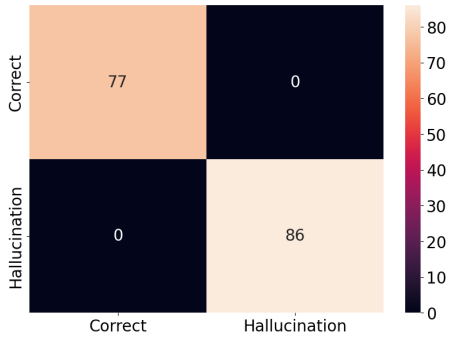


**Figure 2: Confusion matrix of GPT-4o used to detect hallucinations of GPT3.5**

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 [7] 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. GTP-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 [7], can guarantee, in the end, that almost all the raised 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

**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%** |

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. 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.

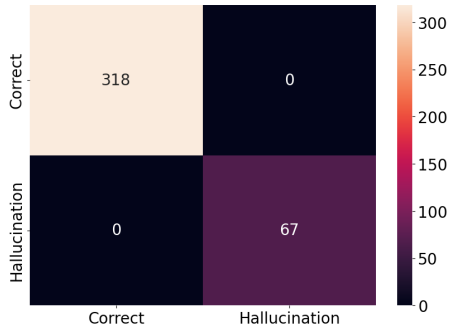


**Figure 3: Confusion matrix of GPT-4o used to detect hallucinations of GPT-4o**

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.

```
<form>
  <h2>Contact Us</h2>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" aria-label="Your name">
  <br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" aria-label="Your email address">
  <br>
  <label for="message">Message:</label>
  <textarea id="message" name="message" aria-label="Your message"></textarea>
  <br>
  <button type="submit">Send</button>
</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.

```
<form action="/submit" method="POST">
  <fieldset>
    <legend>Personal Information</legend>
    <label for="fname">First Name:</label>
    <input type="text" id="fname" name="fname" autocomplete="given-name"
      required>
    <label for="lname">Last Name:</label>
    <input type="text" id="lname" name="lname" autocomplete="family-name"
      required>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" autocomplete="email" required>
  </fieldset>
  <button type="submit">Submit</button>
</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 [28]. 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] AGID - Agenzia per l'Italia Digitale. 2022. Relazione alla Commissione Europea articolo 8, paragrafo 4, della direttiva (UE) 2016/2102. https://www.agid.gov.it/it/design-servizi/accessibilita/monitoraggio

[2] V. Balaji and K.S. Kuppusamy. 2016. Accessibility analysis of e-governance oriented mobile applications. In *2016 International Conference on Accessibility to Digital World (ICADW)*. 141–144. https://doi.org/10.1109/ICADW.2016.7942529

[3] Bundesministerium für Arbeit und Soziales. 2021. Report of the Federal Republic of Germany to the European Commission about the periodic monitoring of compliance with the accessibility requirements of Websites and mobile applications of public bodies pursuant to Article 8 of 2021 Directive (EU) 2016/2102. https://www.bfit-bund.de/DE/Downloads/eu-bericht-pdf.pdf?__blob=publicationFile&v=2

[4] US Congress. 1998. Section 508 of the Rehabilitation Act. https://www.law.cornell.edu/uscode/text/29/794d

[5] Context.ai. 2024. GPT-3.5 Turbo System Card. https://context.ai/model/gpt-3-5-turbo/.

[6] 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. https://doi.org/10.1109/CCNC51664.2024.10454680

[7] 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*.

[8] 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

[9] 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

[10] Mirko Franco, Ombretta Gaggi, and Claudio E. Palazzi. 2023. Analyzing the Use of Large Language Models for Content Moderation with ChatGPT Examples. In *Proceedings of the 3rd International Workshop on Open Challenges in Online Social Networks* (Rome, Italy) *(OASIS '23)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3599696.3612895

[11] 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. https://doi.org/10.1145/3677525.3678693

[12] 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. https://doi.org/10.1109/CCNC49032.2021.9369484

[13] 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. https://doi.org/10.1145/3524458.3547267

[14] World Wide Web Consortium Web Accessibility Initiative Group. 2023. Web Content Accessibility Guidelines (WCAG) 2.2. https://www.w3.org/TR/WCAG22/

[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] Juan-Miguel López-Gil and Juanan Pereira. 2024. Turning manual web accessibility success criteria into automatic: an LLM-based approach. *Universal Access in the Information Society* (March 2024). https://doi.org/10.1007/s10209-024-01108-z

[17] WebAIM Million. 2022. The 2022 report on the accessibility of the top 1,000,000 home pages. https://webaim.org/projects/million/

[18] Sankha Subhra Mullick, Mohan Bhambhani, Suhit Sinha, Akshat Mathur, Somya Gupta, and Jidnya Shah. 2023. Content Moderation for Evolving Policies using Binary Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams (Eds.). Association for Computational Linguistics, Toronto, Canada, 561–573. https://doi.org/10.18653/v1/2023.acl-industry.54

[19] Observatorio de Accesibilidad. 2021. Report on the result monitoring Period 2020-202. https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Accesibilidad/Informe-Resultado-Seguimiento/Resultados-Seguimiento.html

[20] OpenAI. 2024. GPT-4o System Card. https://openai.com/index/gpt-4o-system-card/.

[21] 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. https://doi.org/10.1145/3594806.3596542

[22] 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

[23] 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

[24] 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

[25] United Nations. 2006. Convention on the Rights of Persons with Disabilities.

[26] 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. https://doi.org/10.1145/3643540

[27] Shunguo Yan and P. G. Ramachandran. 2019. The Current Status of Accessibility in Mobile Apps. *ACM Trans. Access. Comput.* 12, 1, Article 3 (feb 2019), 31 pages. https://doi.org/10.1145/3300176

[28] 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