POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Part 2: LLM Based Web Agents: A vertical Literature Review

EXAM OF THE COURSE:
062786 - LARGE LANGUAGE MODELS: APPLICATIONS, OPPORTUNITIES AND RISKS

Giacomo Colosio: 10775899, Tito Nicola Drugman: 10847631

Lecturers:
Carman Mark James,
Brambilla Marco,
Pierri Francesco

Academic year:
2024-2025

Abstract: In this document we present a focused literature review on web agents, a specific class of agents based on Large Language Models (LLMs) and Large Multimodal Models (LMMs) that are capable of interacting with the web. Initially, such agents were primarily designed to retrieve up-to-date information from online sources. However, a recent shift has emerged and now modern web agents are increasingly being developed to perform complex tasks on the web and thus emulating human-like behavior.
Due to time and page constraints, we limited our in-depth literature review to a smaller set of papers. Our opinions and considerations are clearly indicated when presented.

Key-words: Web Agent, Browser Automation, WebGPT, WebArena, WebVoyager, WebOlympus, Large Language Model (LLM), Large Multimodal Model (LMM), Set-of-Mark Prompting.

1. Introduction

Large Language Models (LLMs)—neural networks trained on web-scale corpora—and their multimodal extensions (LMMs) have revolutionized natural-language technology, powering translation and text generation [1, 7], code synthesis [2], vision–language reasoning [18], molecular design [30] and enterprise automation [41]. In their first incarnation these models were *static*: parameters frozen at training time, inevitably drifting out of sync with the ever-changing web. A second wave added read-only internet access—LLMs query a search-engine API, skim snippets and weave up-to-date evidence into an answer. The natural third step is to move from reading to doing: equipping the model to perceive live pages and issue low-level browser actions so that it can complete tasks, not just retrieve facts, within dynamic online environments [12, 44], thereby in a sense mimicking human interaction with the web.

A *web agent* exemplifies this kind of system: it observes the user interface, infers the goal and turns that understanding into a sequence of clicks, keystrokes and navigation that mirror human behavior [38]. Research has split into two broad strands:

- **Agent design.** It explores *how* an LLM should perceive, reason and act in a browser. Work ranges from lightweight *ReAct*-style loops—observe, generate a short chain-of-thought, execute one action—to richer program-generation frameworks that emit an explicit sequence of commands. Other proposals add external memory, tool use or search over alternative action paths; all aim to improve sample efficiency, robustness and interpretability.

- **Evaluation and benchmarking.** Assessing a web agent is inherently complex: success depends not only on producing the right final output but also on interaction efficiency, robustness to dynamic content and graceful failure handling. Dedicated testbeds therefore provide controlled yet increasingly realistic websites, standardized task suites and multi-dimensional metrics that try to capture these aspects.

This *vertical literature review* charts progress in autonomous web agents from 2021 to May 2025. It addresses three guiding questions:

Q1. How has the design of web agents progressed and what common paradigms underpin this evolution?
Q2. What methodological choices in benchmarking and evaluation most influence the conclusions drawn about an agent's effectiveness and robustness?
Q3. Which technical, ethical and safety gaps identified across the literature remain open research challenges for the next generation of web agents?

Section 2 documents the search strategy and inclusion criteria, with a detailed PRISMA flow in Appendix A;
Section 3 traces the chronological evolution of main landmark systems and their evaluation protocols, with a graphical summary in Appendix B;
Section 4 outlines directions for future work;
Section 5 examines the outstanding technical and ethical issues;
Section 6 answer to the initial questions and make a final analysis on the future of this technology.

2. Search Method

We followed the transparent search protocol recommended by Cronin, Ryan and Coughlan [4]. A single pre-registered query was run in four major repositories—arXiv, ACL Anthology, IEEE Xplore and Web of Science—using `("web agent" OR "browser automation") AND (LLM OR GPT)` over the period 2020–2025 and applied to titles, abstracts and keywords. Each record was labelled as one of four types: (1) Web-agent design, (2) Evaluation / benchmark, (3) Paradigm or methodology, (4) Non-web agent. Only types 1 and 2 are analysed in this eight-page review; types 3 and 4 are listed in Appendix A but not discussed further.

The PRISMA diagram that can be see in Table 5 in Appendix 7 shows that 27 records were initially identified; 16 remained after title/abstract screening; 7 primary studies met all inclusion criteria and form the core corpus. A study was retained if it (i) described an LLM-based system that executes low-level browser actions (click, type, scroll, *etc.*) and (ii) reported quantitative results on at least one public benchmark or live-website task. Position papers and theory-only work were excluded. For every included paper we extracted the model architecture, input modality, action space, benchmark(s) used and stated limitations. Grey literature was not searched and backward citation chasing of the six core papers revealed no further eligible studies.

3. Chronological Survey

This section traces the field's evolution through some papers, ordered by year of publication. For each study we briefly summaries the core idea, the action space, the evaluation setting and the main contribution, highlighting how successive works extend (or rethink) the design space of web agents and their benchmarks.

3.1. WebGPT: Browser-assisted question-answering with human feedback [25]

WebGPT consists of a fine-tuned GPT-3 [1] to answer long-form questions using a text-based web-browsing environment which allows the model to search and navigate the web. As the paper highlights, previous work on question-answering such as REALM [11] and RAG [17] focused on improving document retrieval for a given query, instead WebGPT uses a modern search engine (Bing) to browse the internet like a human. WebGPT was trained to answer questions from ELI5 [8] (a dataset of questions taken from a subReddit) and the answers are judged for their factual accuracy, coherence and overall usefulness.
The prompt of the model consists of a written summary of the current state of the environment that includes the question, the text of the current page as well as the current cursor location. The model must decide which of the possible given actions to perform. WebGPT will continue browsing until either the model issues a command to end browsing or the maximum number of actions has been reached or the maximum total length of references has been reached. If at least one reference has been collected, the model is then prompted with the original question and the reference(s), and must compose its final answer.

3.1.1 Actions space

The authors identified a set of actions that the model is allowed to perform (see Table 1). This predefined command set was deemed sufficient for navigating the web. Any output outside of this set is treated as an invalid action.

Command	**Effect**
Search <query>	Send <query> to the Bing API and display a search results page
Clicked on link <link ID>	Follow the link with the given ID to a new page
Find in page: <text>	Find the next occurrence of <text> and scroll to it
Quote: <text>	If <text> is found in the current page, add it as a reference
Scrolled down <1, 2, 3>	Scroll down a number of times
Scrolled up <1, 2, 3>	Scroll up a number of times
Top	Scroll to the top of the page
Back	Go to the previous page
End: Answer	End browsing and move to answering phase
End: <Nonsense, Controversial>	End browsing and skip answering phase

Table 1: Action space of WebGPT.

3.1.2 Environment design

WebGPT operates in a purely text-based browsing environment. When a search is performed, the query is sent to the Microsoft Bing Web Search API. When a link to a new page is clicked, a Node.js script fetches the HTML content of the page and simplifies it using Mozilla's Readability.js [24]. Additionally, elements such as links, images, superscripts, and subscripts are converted into specific text formats. Finally, the remaining HTML is converted to plain text using html2text [34].

3.1.3 Results

The authors decided to evaluate WebGPT on the ELI5 test in two different ways:

- Compare model-generated answers to answers written by demonstrators using the web-browsing environment. The best WebGPT model produces answers that are preferred to those written by the human demonstrators 56% of the time.
- Compare model-generated answers to the reference answers from the ELI5 dataset (which are the highest-voted answers from Reddit). In this case Reddit answers do not typically include citations, thus all citations and references from the model-generated answers were removed. The best WebGPT model produces answers that are preferred to the reference answer 69% of the time, which is a substantial improvement with respect to previous research [16] whose best model's answers are preferred 23% of the time to the reference answers (despite requiring considerably less computational resources than WebGPT smallest model).

3.1.4 Personal Considerations

WebGPT represents a significant milestone in bringing together large-scale language models and real-time web access: by fine-tuning GPT-3 to "browse" and cite sources, it demonstrates that models can learn to navigate search results, extract relevant information and compose well-supported long-form answers. However, its capabilities remain narrowly tailored to question-answering: every possible action (search, click, quote, scroll) of the model was specifically designed to find sources to answer a question and not to carry out richer, multi-step online workflows. To evolve into a truly versatile web agent, future work will need to expand the action space (e.g., filling forms, posting comments, handling authentication or transactions), manage longer-term state or memory across interactions and integrate planning components that can decompose high-level user goals into sequences of web operations, this will require advances in environment design and other processes. We consider WebGPT a passive web agent because its objective is limited to retrieving and summarizing existing content from the web. It does not perform active manipulations or complex interactions with dynamic content, focusing instead on reading and quoting web pages to support long-form answers.

3.2. WebArena: A Realistic Web Environment for Building Autonomous Agents [44]

WebArena is a standalone self-hostable web environment for building autonomous agents, in its original paper is presented to be realistic as well as a reproducible web environment designed to facilitate the development of autonomous agents capable of executing tasks. Its environment comprises four fully operational self-hosted web applications and each represents a distinct domain prevalent on the internet (online shopping, discussion forums, collaborative development and content management) and their content is extracted from their real-world counterparts such as WebShop [37]. Furthermore, it incorporates several utility tools such as map, calculator and scratchpad to best support possible human-like tasks executions. WebArena is also completed with an extensive collection of documentation and knowledge bases from general resources like English Wikipedia and other sources.
WebArena differs from WebGPT in that, while WebGPT uses a search engine (Bing) to browse the internet in a human-like manner, WebArena operates on a fixed set of websites from four popular categories, with functionality and data designed to mimic their real-world counterparts. However, it still represents an oversimplified simulation of real-world web interaction and, therefore, cannot be considered true "internet surfing". As an example, the authors of WebArena constructed a shopping website with approximately 90k products, including prices, options, detailed produced descriptions, images and reviews spanning over 300 product categories with data obtained from actual online sites [37]. WebArena uses the accessibility tree [23], which is a subset of the DOM (Document Object Model) tree, consisting of elements considered relevant and useful for displaying the contents of a webpage. It retains the structured information of the page while being more compact than the full DOM representation.
WebArena includes 812 instantiated tasks derived from 241 templates, designed to be complex, creative and high-level, requiring multiple steps to complete. The templates are flexible, allowing variation through element substitution. WebArena also decided to classify intents in three sub-categories:

- **Information seeking**: which are tasks that expect a textual response that often requires navigation across multiple pages or focus on user-centric content. As an example: *When was the last time I bought shampoo?*
- **Site navigation**: tasks that require navigating through web pages using a variety of interactive elements such as search functions and links with the goal of locating specific information or navigating to a particular section of a site. As an example: *Show me the ergonomic chair with the best rating.*
- **Content and configuration operation**: encapsulate tasks that require operating in the environment to create, revise or configure content or settings. It requires adjusting settings, managing accounts and performing online transactions. As an example: *Post to ask "whether I need a car in NYC".*

3.2.1 Action space

WebArena categorized all the possible keyboard and mouse operations that are available on the web pages in three distinct categories:

- **Element operations**: do nothing, click an element, hover on an element, type to an element, press a key combination and scroll up and down
- **Tab-related actions**: focus on a i-th tab, open a new tab, close current tab
- **URL navigation actions**: visit the last URL, undo visiting the last URL, go to URL

A key distinction between WebArena and WebGPT lies in the richness and granularity of their action spaces. While WebGPT operates within a narrowly defined set of commands tailored specifically for question answering, such as Search, Click, Quote and Scroll, WebArena introduces a significantly more comprehensive and expressive action set. This includes not only standard element-level interactions like clicking and typing but also advanced controls such as multi-tab management, direct URL navigation and interactions grounded in element IDs or screen coordinates. These enhancements support a wider array of task structures, enabling agents to perform more realistic and diverse web-based operations beyond fact-finding.
With WebArena it is possible to select an element by its on-screen coordinates or by a unique ID that is assigned to each element. The ID is generated when traversing the DOM or accessibility tree. Thanks to the element IDs, the element selection is transformed into an n-way classification problem.

3.2.2 Results

The best model (GPT-4 [26]) with CoT prompting achieves a modest end-to-end task success rate of 11.70%, which is significantly lower than the human performance of 78.24%. Interestingly, it was observed that, among the tested models, there was an early stopping due to the model's conclusion of unachievability, as an example GPT-4 erroneously identified 54.9% of feasible tasks as impossible. The authors argued that this was due to a

hint in the instruction which asked the model to stop when encountering unreachable questions, so this made the models over-cautions and sometimes quitting on tasks that were actually doable. When the hint was removed it led to a performance boost in achievable tasks, enhancing the overall task success rate of GPT-4 to 14.41%, better but still very far from the human performance.

3.2.3 Conclusion

An important difference with respect to the past research is that WebArena was the first web environment to consider multi-tab web-based tasks to promote tool usage, direct comparison and reference across tables and other functionalities. This offers a more authentic replication of human web browsing habits compared to maintaining everything in a single tab. This paper moved the web-agent one step forward since it found out that telling the model to stop if it believed the task to be unachievable made them become overly conservative, often misclassifying feasible tasks as impossible and stopping prematurely. Nevertheless, even if WebArena made an effort to reduce simplifications to the minimum, it still does not focus on a real-world simulation. The authors argued that to achieve reproducibility of their result it was necessary to make the environment standalone and thus not relying on live websites. In this way they were able to circumvent technical challenges such as bots being subject to CAPTCHAs and unpredictable content modifications and configuration changes which obstruct a fair comparison across different systems over time. However, this also means WebArena is a curated, static slice of the internet, sidestepping real-world challenges such as dynamic content loading, cookies, authentication barriers (e.g., multi-factor login), session handling, personalized recommendations and the presence of noisy or adversarial user-generated content.
Despite the strength of models like GPT-4, WebArena reveals key limitations in observation interpretation. Specifically, agents often overlook subtle but critical interface details, such as previously entered input or recent actions. Leading to redundant or failed behaviors. The authors suggested that these observed failures are related to the current pretraining and supervised fine-tuning on dialogues employed in GPT models [28] where immediate context is prioritized and small variations are often ignored.
One notable aspect of WebArena is that, in addition to the accessibility tree, the authors provide flexible configurations to render the web page content in various modalities. These include the raw HTML rendered as a full DOM tree, as commonly used in prior work [5, 19, 31] and a screenshot of the page represented as an RGB image. While these options are available, comparing the effectiveness of different modalities was beyond the scope of their study.

3.3. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models [12]

WebVoyager starting point is that vision capabilities are crucial for utilizing tools such as web browsers, since web pages are designed with user experience emphasizing intuitive information and structured presentation and it is more effective than mere HTML representation. Furthermore LMMs demonstrated a remarkable ability to integrate visual cues with textual information [10, 27]. WebVoyager is a multi-modal web agent designed to autonomously accomplish web tasks online without any intermediate human intervention. WebVoyager processes the user query by making observations from screenshots and textual content in interactive web elements, formulates a thought on what action to take (such as clicking, typing, scrolling, etc.) and then executes that action on the websites.
An interesting result is that marking clickable web elements (i.e., overlaying numbered labels on interactive elements such as buttons, input fields and links) not only facilitates decision-making in web agents but significantly improves their task completion performance [36]. This technique provides explicit spatial and functional clues that help agents disambiguate dense or visually complex web layouts, allowing the agent to more accurately localize relevant elements and determine appropriate actions.
A first and important difference with WebArena is that WebVoyager does not host any website locally and allows, the agent to explore the open web instead, which poses unique challenges such as floating ads, pop-up windows and many more. Nevertheless, the authors, due to technical limitations, omit using website requiring login or CAPTCHA to access their content. They selected 15 representative websites that cover different aspects to ensure diversity in the evaluation and they also decided to include Google Search, which is a universal website that can serve as a starting point for any website. In total the authors collected 40+ low repetition tasks per website, for a total of 643 tasks. At every step the agent receives the current screenshot, auxiliary text and history as input.

3.3.1 Action Space

WebVoyager decided to implement the most commonly used mouse and keyboard actions which ranges from basic web navigation to more complex operations tailored to efficiently gather data and respond to queries. The list of commands can be seen in Table 2. Notice that the opposite of the Back action is unnecessary since it can be achieved by repeating previous actions.

Action Name	Command Format	Description
Click	`Click [Numerical_Label]`	Click on an interactive element (e.g., a button or link) identified by the numeric label.
Type	`Input [Numerical_Label]; [Content]`	Select a text box, deletes existing content, inputs new content and presses Enter.
Scroll	`Scroll [Numerical_Label or WINDOW]; [up or down]`	Scroll the whole page or a specific scrollable area in the indicated direction.
Wait	`Wait`	Pause to allow time for the webpage or elements to load.
Back	`GoBack`	Return to the previous webpage.
Jump to Search Engine	`Google`	Redirect the agent to Google Search when stuck or no answer is found.
Answer	`ANSWER; [Content]`	End the task and submits the final answer.

Table 2: WebVoyager Action Space.

3.3.2 Results

WevVoyager was able to outperform text-only GPT-4 (All Tools) baselines by large margins in most websites tasks: the succes rate was 59.1% for WebVoyager and 30.8%. It is interesting to notice that for some tasks WevVoyager performs slightly lower than the Text-only version. The authors argue that this happened on websites that are more text-heavy than others since WebVoyager relies on web screenshots for decision making, dense text might not be easily recognizable from the image. WebVoyager also had a 30% success rate on SeeAct [42] online test set whereas the best SeeAct autonomous agent has 26%. On the benchmark proposed by the authors of WebVoyager, WebVoyager itself achieves a 59.1% task success rate significantly surpassing the performance of both GPT-4 (All Tools) and the WebVoyager (text-only) setup.

3.3.3 Conclusion

The core idea behind WebVoyager—relying on online interactions with real websites—more accurately reflects real-world use cases. A successful web agent should be able to adapt to these challenges and consistently solve tasks in such dynamic environments.
WebVoyager empirically selects black for the borders and background of the labels to enhance clarity. Even if the authors observed that using a single black color yields higher success rates than using multiple colors, it would be interesting to explore alternative designs, such as applying a grayscale filter to the entire webpage while keeping the labels and borders in color to improve visual contrast. This could also be a method to solve a issue reported by the authors which found out that the agent sometimes, due to proximity, selects the wrong element. As an example, sometimes it confuses adjacent elements and misinterprets numbers on a calendar as numerical labels. Some similar test results are avaible on Part I.
As highlighted before on text-heavy websites WebVoyager performed not as good as the Text-only counterpart, the authors suggested that extracting such dense text from the HTML to augment the input could be a potential solution to this problem.
WebVoyager authors highlighted that the most common failure was running out of steps before completing the task. This happened mainly in three scenarios: (i) when the agent search query is not precise enough (the model becomes overwhelmed by irrelevant search results), (ii) when the scroll-able area is very small and lastly (iii) sometimes in the middle of the page the agent has trouble deciding whether to scroll up or down.

3.4. WebOlympus: An Open Platform for Web Agents on Live Websites [43]

WebOlympus is an open platform designed to foster the research and deployment of web agents on live websites. The agent system accepts observations from the website and generates grounded actions to execute on the website. At each step in the sequence, WebOlympus must generate an action description based on prior actions,

the current state and previous observations. Since past research has utilized different styles of observation spaces—such as raw HTML [5], screenshots [12, 42], DOM trees and accessibility trees—WebOlympus supports all these input modalities as contextual information. A distinctive feature of WebOlympus is its user-friendly web interface, provided via a Chrome browser extension, which allows users to directly interact with the web agent. The interface can be used for:

- **Task control**: allows the user to start the agent and terminate the task during execution.
- **Action visualization**: display the intermediate processes of the agent executing the task. It shows both previous action performed by the agent as well as the next step the agent has generated before its execution.
- **Monitor mode**: the user can accept or reject an agent action before the execution. It also allows to send messages to the agent.
- **Trajectory recording**: allows the user to review the entire execution trajectory.

3.4.1 Action Space

WebOlympus action space allows the agent to take a wild range of actions that we reported in Table 3. Similarly to WebArena [44] the authors decided to classify the actions in three groups: (1) single-page actions, (2) multi-tab operations and (3) inter-page navigation activities.

Action	Description
Click (elem)	Click on a webpage element using the mouse.
Hover (elem)	Hover the mouse over an element without clicking it.
Select (elem)	Choose an option from a selection menu.
Type (elem, text)	Enter text into a text area or text box.
Enter	Press the Enter key, typically to submit a form or confirm an input.
Scroll	Scroll the webpage up or down by half of the window height.
Close_tab	Close the current tab in the browser.
Open_tab	Open a new tab in the browser.
Go_forward	Navigate to the next page in the browser history.
Go_back	Navigate to the previous page in the browser history.
Goto (URL)	Navigate to a specific URL.
Say (text)	Output answers or other information the agent wants to tell the user.
Memorize (text)	Keep some content in action history to memorize it.

Table 3: Actions space of WebOlympus.

3.4.2 Results

For the evaluation, WebOlympus followed the strategy proposed by SeeAct [42], randomly sampling 50 tasks from Mind2Web [6] to be executed on live websites. The MindAct agent [6], based on FLAN-T5-XL [3] and fine-tuned on the Mind2Web training data, achieved a success rate of 16%. In contrast, GPT-4, without any additional fine-tuning on Mind2Web, achieved a success rate of 22% on the same tasks. The SeeAct agent [42] achieved a 48% success rate using a textual-choice grounding method (framing the task as a multiple-choice question and asking the model to select one option) and a 56% success rate using a Set-of-Mark grounding method (rendering the screen as an image, tagging each clickable element with a visual label and asking the model to choose among them).

3.4.3 Conclusion

WebOlympus is one of the few papers we encountered that explicitly addresses the important issue of safety, as discussed in Section 5. To mitigate potential risks, WebOlympus incorporates a safety monitor module that detects state-changing actions and forwards potentially risky ones to the user for approval [15, 42]. While the safest approach would be to forward all actions to the user before execution [42], this is neither practical nor consistent with the goal of building autonomous agents. To balance safety and autonomy, the authors designed a classifier based on GPT-4V [42] capable of identifying a subset of state-changing actions. To evaluate the

safety monitor, the authors annotated 48 state-changing and 108 non-state-changing actions. The performance results are summarized in Table 4.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP) = 64	False Negative (FN) = 5
Actual Negative	False Positive (FP) = 44	True Negative (TN) = 43

Table 4: Confusion matrix for safety monitor performance of WebOlympus.

4. Future Work

There are many steps that still needs to be performed to extend web-agents. All the papers we analyzed supported some actions which are only a subset compared to all the actions a human user can take while browsing the web (e.g. drag action). We saw from this point of view some improvements such from WebGPT to WebArena, but the journey is still long.
Additionally, from all the papers we review only WebArena [44] and WebOlympus [43] decided to deal with mutli-tab functionality, which we believe is truly more representative of how humans usually surf on the internet. Nevertheless, we find no comprehensive study that directly compares the performance of web agents operating in multi-tab versus single-tab environments.
Many papers analyze different basic file formats such as PDF [12, 25] and text files [12], but it was not always granted [44]. Supporting popular formats such as PDFs would be highly beneficial, as a study by Phil Ydens (Adobe's Vice President of Engineering for Document Cloud) in 2015 estimated that there were almost 2.5 trillion PDF documents worldwide [40] and it is safe to assume that this number has increased significantly since then. Enhancing support for additional more complex file formats is a crucial step in the development of web agents and we believe an important format is video (which could be implemented to start with the transcript available on Youtube). Furthermore, solving CAPTCHAs appears to be outside the scope of most web agent research. However, recent studies have shown that current AI technologies can exploit advanced image-based CAPTCHAs [29].

5. Ethical and Safety Considerations

WebGPT's authors highlighted that since WebGPT has live access to the web via the text-based browsing environment it could potentially pose risks to both the user and to others. As an example, the model could edit Wikipedia to construct a reliable-looking reference. The authors also noted their belief that the risks posed by WebGPT through the exploitation of real-world side effects are very low, as its only interactions with the external world are limited to sending queries to the Bing API and following existing web links. Actions such as editing Wikipedia are not directly accessible to the model.
By curating the environment, WebArena can filter out toxic, misleading or adversarial content, but this also means it doesn't expose agents to the ethical complexities of moderating or navigating such content in the wild. The authors justify their controlled setup as necessary for scientific reproducibility, but we can discuss if those models are truly ready for the open internet where stakes, ambiguity and harm potential are higher.
It is important to underline that web agents operating without restrictions can pose safety risks. A critical concern highlighted [43] is that web agents may perform state-changing actions that alter the state of the website in a hard to-reverse and undesirable way. As an example, an agent can complete the task of "*scheduling a Model 3 demo drive at Tesla*" and in the final step, the agent will click the "*Schedule Demo Drive*" button: this action's impact is irreversible, as it sends a demo drive request directly to the website server.
To enable web agents to operate smoothly and safely on live websites, a method to automatically identify risky actions is necessary [15, 42]. Some solutions [43] suggested to use implement a classifier to identify some state-changing actions, but as the authors highlighted it does not perfectly ensure safety.
Web agents can pose concerns related to privacy issues, such as access to users' personal profiles and sensitive operations, including financial transactions and application form submissions. There is also the possibility for web agents to generate harmful actions on the web that can cause irreversible changes to the website state [43].

6. Conclusion

A recurring theme across many papers is the adoption of a step-by-step, sequential decision-making approach, where agents choose actions based on current observations rather than planning the entire sequence in advance. Existing works [13, 18, 22] observe that LLMs could break a task into more manageable sub-tasks [46]. This strategy seems to mirror how humans often navigate the web without a precise roadmap, but with a general goal and the flexibility to adapt based on what they see. Sequential planning not only enables greater adaptability and robustness, especially when interacting with unpredictable or unfamiliar web interfaces, but also allows agents to backtrack or revise their course if they encounter dead ends or misleading content and thus introduce a more structured approach to planning while also allowing for decision reconsideration [21, 39]. This makes the approach both practical and powerful, particularly in real-world settings where deterministically mapping all possible navigation paths is infeasible. Existing work also incorporates failure recovery, self-correction [14, 32] and observation summarization [33] to improve execution robustness and allowing the agent to reflect on failed actions, adjust reasoning and retry but also help agents to operate under limited context by summarizing long interaction histories. Some researchers seem to highlight that breaking a task into smaller sub pieces could be further refined by representing task executions as programs. This approach enables more effective sub-task management and skill reuse [9, 20, 35, 45], as it allows agents to construct structured, interpretable programs that define a coherent sequence of steps needed to complete the task.

7. Bibliography and citations

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint*, arXiv:2005.14165, 2020.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, and et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[3] Hyung Won Chung, Le Hou, Shixiang Shane Gu, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Samuel Longpre, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *arXiv preprint*, arXiv:2210.11416, 2022.

[4] Patricia Cronin, Frances Ryan, and Michael Coughlan. Undertaking a literature review: a step-by-step approach. *British Journal of Nursing*, 17(1):38–43, 2008.

[5] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint*, arXiv:2306.06070, 2023.

[6] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint*, arXiv:2306.06070, 2023. NeurIPS 2023 Spotlight, version 3 (Dec 9, 2023).

[7] Angela Fan, Shruti Bhosale, Holger Schwenk, and et al. Beyond english–centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48, 2021.

[8] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long form question answering. *arXiv preprint*, arXiv:1907.09190, 2019.

[9] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10764–10799. PMLR, 2023.

[10] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, and et al. Gemini: A family of highly capable multimodal models. *arXiv preprint*, arXiv:2312.11805, 2023.

[11] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR, 13–18 Jul 2020.

[12] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint*, arXiv:2401.13919, 2024. Accepted to ACL 2024.

[13] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147, Baltimore, Maryland, USA, July 17–23 2022. PMLR.

[14] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *arXiv preprint*, abs/2303.17491, 2023.

[15] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv preprint*, arXiv:2407.01476, 2024.

[16] Kalpesh Krishna, Arvind Neelakantan Roy, and Mohit Iyyer. Hurdles to progress in long-form question answering. *arXiv preprint*, arXiv:2103.06332, 2021.

[17] 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. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint*, arXiv:2005.11401, 2020. Accepted at NeurIPS 2020.

[18] Xinze Li, Yixin Cao, Muhao Chen, and Aixin Sun. Take a break in the middle: Investigating subgoals towards hierarchical script generation. *arXiv preprint*, abs/2305.10907, 2023.

[19] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8198–8210, Online, 2020. Association for Computational Linguistics.

[20] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[21] Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint*, abs/2305.08291, 2023.

[22] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1384–1403, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics.

[23] MDN Web Docs. Accessibility tree - mdn web docs. `https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree`.

[24] Mozilla. Readability. `https://github.com/mozilla/readability`, 2024. Accessed: 2025-05-26.

[25] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint*, arXiv:2112.09332, 2021.

[26] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[27] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and et al. Gpt-4 technical report. *arXiv preprint*, arXiv:2303.08774, 2023. Version 6.

[28] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.

[29] Andreas Plesner, Tobias Vontobel, and Roger Wattenhofer. Breaking recaptchav2. *arXiv preprint*, arXiv:2409.08831, 2024. Accepted at COMPSAC 2024.

[30] Alexander Rives, Joshua Meier, Tom Sercu, and et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[31] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144, Sydney, NSW, Australia, 2017. PMLR.

[32] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: An autonomous agent with dynamic memory and self-reflection. *arXiv preprint*, abs/2303.11366, 2023.

[33] Abishek Sridhar, Robert Lo, Frank F. Xu, Hao Zhu, and Shuyan Zhou. Hierarchical prompting assists large language model on web navigation. *arXiv preprint*, arXiv:2305.14257, 2023.

[34] Aaron Swartz. html2text: Convert html to markdown-formatted text. `https://github.com/aaronsw/html2text`, 2025. Accessed: 2025-05-29.

[35] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint*, abs/2305.16291, 2023.

[36] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint*, arXiv:2310.11441, 2023.

[37] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *arXiv preprint*, arXiv:2207.01206, 2022. NeurIPS 2022 camera-ready version, v4.

[38] Shunyu Yao, Dian Yu, Jeffrey Zhao, and et al. React: Synergizing reasoning and acting in language models. In *Advances in Neural Information Processing Systems*, 2023.

[39] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint*, abs/2305.10601, 2023.

[40] Phil Ydens. Day 1 keynote - phil ydens. YouTube video, PDF Association, 2023. Accessed: 2025-05-26.

[41] Lei Zhang, Jinjun Xiong, and Luca Benini. Large language models for robotic process automation. *IEEE Transactions on Emerging Topics in Computing*, 2022.

[42] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint*, arXiv:2401.01614, 2024.

[43] Boyuan Zheng, Boyu Gou, Scott Salisbury, Zheng Du, Huan Sun, and Yu Su. WebOlympus: An open platform for web agents on live websites. In Delia Irazu Hernandez Farias, Tom Hope, and Manling Li, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 187–197, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[44] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint*, arXiv:2307.13854, 2023.

[45] Shuyan Zhou, Pengcheng Yin, and Graham Neubig. Hierarchical control of situated agents through natural language. In *Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI)*, pages 67–84, Seattle, USA, 2022. Association for Computational Linguistics.

[46] Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. Show me more details: Discovering hierarchies of procedures from semi-structured web data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2998–3012, Dublin, Ireland, 2022. Association for Computational Linguistics.

Appendix A — PRISMA Flow and Corpus

A.1 PRISMA flow

Phase	Description	Records
Identification	Search 2019–2025 in arXiv, ACL Anthology, IEEE Xplore, Web of Science with `("web agent" OR "browser automation") AND (LLM OR GPT)`.	27
Screening	Title/abstract check. Non-web agents and paradigm-only papers removed.	−16
Eligibility	Full-text check. Workshop duplicates, dataset-only papers, no-result studies removed.	−7
Included	Design **A** / Benchmark **B** papers most cited in field.	**4**

Table 5: PRISMA flow.

Note. The review is restricted to 8 pages; therefore grey literature (e.g. company blogs) was not searched systematically. Backward citation chasing of the included papers yielded no further eligible primary studies.

A.2 Final corpus (ordered by year)

Title	First author, year	Cat.†
WebGPT	Nakano, 2021	A
WebArena	Zhou, 2023	A+B
WebVoyager	He, 2024	A
WebOlympus	Zheng, 2024	A

Table 6: Primary studies analysed in this review.

†Category: **A** = web-agent design paper; **B** = benchmark / evaluation paper.

†Category: **A** = Web-agent design; **B** = Evaluation/benchmarking ; **C** = Excluded categories (non-web agents and paradigm papers).

A.3 Complete set of 27 screened records

#	Title (first author, year)	Cat.†
1	*WebGPT* (Nakano, 2021)	A
2	*ReAct* (Yao, 2022)	C
3	*Mind2Web* (Deng, 2023)	B
4	*Language Models Can Solve Computer Tasks* (Kim, 2023)	C
5	*WebWISE* (Tao, 2023)	A
6	*Real-World WebAgent* (Gur, 2024)	A
7	*AutoWebGLM* (Lai, 2024)	A
8	*WebVoyager* (He, 2024)	A
9	*GPT-4V Grounded / SeeAct* (Zheng, 2024)	A
10	*WebArena* (Zhou, 2023)	B
11	*SeeAct Online Suite* (Zheng, 2024)	B

12	*Steward* (Tang, 2024)	A
13	*ScreenAgent* (Niu, 2024)	C
14	*AgentOccam* (Yang, 2024)	B
15	*Agent-E* (Abuelsaad, 2024)	A
16	*Falcon-UI* (Shen, 2024)	C
17	*ShowUI* (Lin, 2024)	C
18	*ClickAgent* (Hoscilowicz, 2024)	A
19	*NaviQAte* (Shahbandeh, 2024)	A
20	*ScribeAgent* (Shen, 2024)	A
21	*OmniParser* (Lu, 2024)	C
22	*AGUVIS* (Xu, 2024)	C
23	*AutoGLM* (Liu, 2024)	C
24	*MMAC-Copilot* (Song, 2024)	C
25	*OS-Copilot* (Wu, 2024)	C
26	*GPT-4V is a Generalist Web Agent* (Zheng, 2024)	A
27	*Plan-and-Act* (Erdogan, 2025)	C

[†]Category legend: **A** = web-agent design; **B** = benchmark / evaluation; **C** = paradigm / methodology or non-web agent.

Appendix B – Synthesis of Key Features Across Web Agent Models

In this section, we provide a side-by-side comparison of the main features of WebGPT [25], WebArena [44], WebVoyager [12] and WebOlympus [43]. These tables are intended to distill the most critical differences and trends across the evolution of web agents.

Category	WebGPT	WebArena	WebVoyager	WebOlympus
Navigations	Search via Bing API	Go to URL Visit/undo visiting the last URL	Jump to search engine	goto (URL) go_forward, go_back
Actions	Click links Scroll up/down Jump to top Find in page Go back to previous page	Click an element Hover on an element Type to an element Press a key combination Scroll up/down	Click Type (input) Scroll up/down Back Wait	Click / Select Hover Type Enter Scroll
Multi-tab		Focus on a specific tab Open/close a new tab Do nothing		Open tab / Close tab

Table 8: Detailed comparison of action spaces across WebGPT, WebArena, WebVoyager and WebOlympus.

Modality	WebGPT	WebArena	WebVoyager	WebOlympus
Plain text (simplified HTML)	Converts HTML to text	Offers accessibility tree; optional raw HTML	Uses auxiliary text, but not as main input	Provides raw HTML
HTML DOM tree	Not used	Available as an optional modality	Not used	Available as an optional modality (via HTML→DOM conversion)
Accessibility tree	Not used	Default representation	Available in text-only version	Available as an optional modality (via HTML→accessibility conversion)
Screenshot / RGB image	Not used	Offered as an alternative modality	Primary input with clickable labels	Offered as a modality (provides raw screenshots)
Multimodal vision + text	Not supported	Limited/optional use	Core approach: fuses vision and text	Core approach: fuses HTML and screenshots

Table 9: Web observation modalities in WebGPT, WebArena, WebVoyager and WebOlympus.

Aspect	WebGPT	WebArena	WebVoyager	WebOlympus
Environment Type	Text-based browser	Self-hosted websites	Live websites	Live websites (Chrome-extension based)
Realism	Medium-low — live pages but rendered as plain text	Medium (realistic content but static)	High (real-time, dynamic content)	High (real-time, dynamic content)
Diversity of Websites	Open Web (via Bing)	4 domains (E-commerce, social forums, collaborative software development, CMS)	15 real websites including Google	Arbitrary live websites (user-provided URLs)
CAPTCHA / Login	Not handled	Avoided for reproducibility	Avoided due to technical limits	Not handled

Table 10: Comparison of environment types and realism across WebGPT, WebArena, WebVoyager and WebOlympus.

Component	WebGPT	WebArena	WebVoyager	WebOlympus
LLM used	GPT-3 (fine-tuned)	GPT-4 (CoT prompting)	LMM (GPT-4 Turbo with vision)	Unified interface via LiteLLM adaptor (OpenAI, Gemini, Anthropic) and local hosting via Ollama.
Planning Approach	Learns a policy via behaviour-cloning + PPO/Rejection-sampling	CoT prompting	observation + thought + action	Flexible language-agent-driven sequence generation; supports various planning strategies.
Memory/Context	Summary of the current state of the environment + question + text of the current page + current cursor location + other information	Intent and the previously performed action	Screenshot + auxiliary text + history. Remove outdated web page information. Keep the three most recent observations in the inputs and the entire history of thoughts and actions	Maintains full execution trajectory (screenshots, HTML, actions) to provide context to the agent.

Table 11: Comparison of agent architecture and reasoning across WebGPT, WebArena, WebVoyager and WebOlympus.