Intelligent Agents: Where Is the Web?

Victor Charpenay[1]

[1]Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS Saint-Etienne, France

Abstract

This paper reviews agent frameworks and benchmark datasets for software agents evolving in a Web environment. All frameworks and datasets make the assumption that agents interact with their environment through text and images alone, hiding the main abstractions of the Web and its architectural constraints. This study raises the question whether software agents that leverage structured data instead, mostly available in an RDF format, can be more accurate or more computationally efficient than agents based on chatbots.

Keywords

Web Agent, Chatbot, Hypermedia, RDF

1. Introduction

"Where are all the intelligent agents?" is the question Jim Hendler asked in 2007 to the Semantic Web community [1]. With the rapid growth of the Web as a collection of interlinked resources, six years earlier, he had envisioned a Web that is not only meant for humans but also for software agents, capable of discovering and using Web services on behalf of humans [2]. In six years, though, no intelligent agents had been developed for the Web and the question could have remained rhetorical if recent progress in natural language processing and computer vision had not reshaped the field.

In the past year, Anthropic and OpenAI introduced specialized chatbots, respectively called Computer Use and Operator, that can autonomously operate a Web browser [3, 4]. The versatility of these chatbots suggests that Hendler's question has finally been answered: here are intelligent agents. However, what immediately follows is another important question: "Where is the Web?".

The architecture of the Web relies on three pillars: identification, interaction and representation, materialized through URIs, HTTP, HTML and other hypermedia formats [5]. Agent frameworks that are based on chatbots largely ignore these three pillars and expose agents to higher abstractions instead, such that agents cannot discern whether they evolve in a desktop environment or a Web environment. Arguably, most humans browse the Web without being aware of its abstractions. But imitating humans in their browsing behavior is a stimulating, yet unreasonable challenge. Much of the processing performed by Web browsers is necessary for agents equipped with keyboard and mouse, not for software agents which only need to know what resources are managed by a server (through links) and how to manipulate their representation (through forms).

This paper reviews frameworks to turn chatbots into Web agents (Section 2) and benchmark datasets featuring tasks to perform on the Web (Section 3). It then introduces criteria to discriminate tasks that are specific to the Web from others (Section 4). Most benchmark datasets, despite targeting Web environments, do not satisfy these criteria.

2. Agent Frameworks

It is straightforward to build an agent from a chatbot. Instead of generating a plausible answer to a question, the chatbot only has to generate a function call, either in a pre-defined format [6] or as a code snippet [7]. A dedicated software component, acting as an intermediary between the agent and its

The Second International Workshop on Hypermedia Multi-Agent Systems (HyperAgents 2025), in conjunction with the 28th European Conference on Artificial Intelligence (ECAI 2025); October 26, 2025, Bologna, Italy

victor.charpenay@emse.fr (V. Charpenay)

0000-0002-9210-1583 (V. Charpenay)

Table 1
Popular agent frameworks based on the ReAct design pattern (column 'Count' gives the number of tools related to the Web over the total number of tools built into the framework)

Framework	Maintainer	Count	Example Web tools
LangGraph	LangChain	26/35	Google search, Tavily search, Slack, Hyberbrowser, …
CrewAI	CrewAI	21/52	Google Serper, Firecrawl, Selenium, Website RAG search, …
AG2	Microsoft, UPenn	9/15	Google search, Perplexity search, Firecrawl, deep research, …
Smolagents	HuggingFace	5/9	Google search, API Web search, visit Web page, …
Agents SDK	OpenAI	1/7	Web search

environment, calls the function and informs the chatbot of possible effects in a textual form. The agent is deemed intelligent if it is capable of anticipating the effects of an action and planning several steps ahead. Such reasoning materializes as an intermediary step in which the chatbot generates free text. Chatbots that combine reasoning and acting are referred to as ReAct agents [6]. With the development of multimodal chatbots, ReAct agents evolved into SeeAct agents to also perceive images [8]. The virtual assistants that Anthropic and OpenAI commercially provide, which they call computer-using agents, are SeeAct agents [3, 4].

Agents based on a chatbot achieve satisfactory results on various tasks without any fine-tuning. They are typically given a system prompt that describes available actions in natural language, at initialization time. For more complex scenarios, such a prompt can be reformulated at run time based on signals received from the environment [9]. If available, the language model underlying the chatbot may even be fine-tuned to generate more accurate function calls. WebGPT, for instance, results from fine-tuning GPT-3 to leverage Web search in question answering tasks [10]. Web search, as we will see later, is one of the most common capabilities of ReAct agents. OpenAI recently integrated it into ChatGPT in a mode called deep research [11].

Actions are made available to chatbots via tools, i.e. pieces of code that easily integrate with the rest of a program [12, 13]. Many tools are defined as a single function, typically in Python, such that a system prompt describing the tool can be automatically derived from its signature and documentation. Many agent libraries are based on this principle, offering users the possibility to create custom tools. However, common tools are often built into the library.

Table 2 gives a summary of Web tools provided by five well-known agent libraries. All libraries provide a Web search tool. All but one give access to more than one search engine (the exception being the OpenAI Agents SDK). Google search is the most frequent tool but other tools appear several times, including Brave search and SearxNG (a meta-search engine), as well as Firecrawl and Hyperbrowser. These two utilities, among others, provide high-level representations of Web pages and Web sites, hiding certain aspects of Web interactions such as CSS rendering and link following. The tools presented in the table can be broadly classified into five categories: search, scraping and crawling, browser automation, platform access, direct access.

Search. Search tools provide search engine response pages (SERPs) in a semi-structured way. Tools exist for traditional search engines (Bing, Brave, DuckDuckGo, Google, Mojeek) but also for more recent search engines dedicated to chatbots (Tavily, You.com, EXA, Linkup, Perplexity). The latter expect full questions as queries (rather than keywords) and use themselves language models to generate a description of each result page.

Scraping and crawling. Scraping consists in extracting structured data from a Web page, possibly after a rendering step. Crawling consists in following links automatically to produce a condensed view of the visited Web pages (for instance, all pages of a Web site). The two procedures are often implemented together in a program, sometimes called crawler or spider. Crawler tools in agent frameworks (Jina, AgentQL, Hyperbrowser, MultiOn, Firecrawl, ScrapFly, ScrapGraph, Browserbase, Crawl4AI) tend to be dedicated to chatbots. Many programs already existed for scraping and crawling but, as for search, recent ones use language models to restructure or simplify Web pages. Some crawling programs also provide deep research functionalities, like ChatGPT.

Browser automation. Scraping programs often operate on dynamic Web pages, requiring a browser environment. To do so, they use a browser automation program that sends key press and click events to a headless browser. Such browser automation programs (Playwright, Selenium, StageHand) can also be directly used as tools.

Platform access. Many platforms primarily accessible via a Web API have a dedicated tool (Github, Gitlab, Gmail, Infobip, Jira, Office365, Slack, Twilio, YouTube, Wikipedia). Messages exchanged between the agent and the platform are formatted in JSON.

Direct access. The remaining tools expose lower-level features, such as HTTP request helpers.

All agent frameworks, except the OpenAI Agents SDK, put Web tools forward. Web tools represent 40%-74% of the built-in tools they provide (other tools give access to files and databases, for instance). However, these tools hide the Web from agents more than they expose to it. Technical details of identification, interaction and representation, the main architectural features of the Web, are indeed invisible to agents.

Identification. Crawlers that aggregate information from several pages within a Web site make invisible what entities are being identified and how they are interlinked. Yet, links between resources may themselves have a meaning—an assumption that is at the foundations of the Semantic Web.

Interaction. Interoperability between clients (agents) and servers (platforms) is not guaranteed by a standard protocol. Instead, the trend to build "walled garden" over Web standards is strengthened, each platform requiring a dedicated tool with its own abstractions. Among other standards, OpenSearch could be an efficient substitute to search tools. Github, YouTube and Wikipedia (at least) follow this standard already.

Representation. Scrapers alter the original representation of pages, generating accessibility trees or textual summaries that are non-standard. Direct content negotiation with servers could be used instead, to provide machine-readable or purely textual representations of HTML pages. A current proposal is to serve MarkDown to software agents as an alternative to HTML[1].

Human agents require a browser to access Web resources. Browsers follow certain links automatically (pointing to CSS and JavaScript resources, among others), pages can be rendered in different ways (Firefox has a reading mode, for instance[2]) and many browser extensions can dynamically alter interactions or representations. Arguably, Web tools provided by agent frameworks collectively play the same role as a browser. However, as they are currently implemented, these tools are inefficient intermediaries. Origin servers typically generate HTML from data that is already (partly) structured; they would more efficiently serve a representation that chatbots can process (after content negotiation). Moreover, most tools cited here are only accessible via a remote service: the agent thus entirely depends on another server that becomes a centralization point, against the original design objective of the Web as a massively distributed information system.

Tools that rely on Web technologies can surely be considered as Web tools. However, agents that use these tools are not exactly Web agents if the main abstractions of the Web (URIs, HTTP, HTML and other hypermedia formats) are not exposed to them.

3. Benchmark Datasets

WebGPT was tested on the ELI5 dataset, extracted from the "explain like I'm five" topic on Reddit [19]. The ReAct approach to software agents was initially evaluated in the WebShop environment, an e-commerce platform populated with information extracted from Amazon [16]. In addition to these two, several benchmark datasets were introduced to evaluate the behavior of agents on the Web, on different aspects. Two families of datasets can be identified: some datasets require interaction with Web servers, others do not except for search.

Datasets that only require search include ELI5, SimpleQA [20], GPQA [21] and GAIA [22]. They are question answering datasets designed to quantify the information retrieval capabilities of agents: agents

[1] see the recent llms.txt proposal.

[2] the underlying JavaScript library, Readability.js, was even used to train WebGPT.

Table 2
Benchmark datasets used to evaluated agents in a Web environment

Dataset	Percepts	Actions
QAWoB [14]	RGB image, DOM	select date, drop down, scroll, click
FormWoB [14]	RGB image, DOM	click, drag, scroll, copy, paste, select
MiniWoB++ [15]	RBG image, DOM	move, click, drag, scroll, copy, paste, select
WebShop [16]	page type, text	search, choose
Mind2Web [17]	RGB image (*optional*), HTML	click, select
WebArena [18]	RGB image, HTML, text	click, scroll, go back/forward, go to

Table 3
Most frequent Web sites used as sources in various question answering datasets: SimpleQA (evaluation split), GPQA (diamond split) and GAIA (not all tasks reviewers explicitly mentioned their sources, figures are likely underestimated)

SimpleQA	wikipedia.org (40%)	fandom.com (3%)	…	
GPQA	wikipedia.org (18%)	nih.gov (10%)	libretexts.org (8%)	stackexchange.com (3%)
GAIA	wikipedia.org (18%)	youtube.com (5%)	github.com (3%)	researchgate.net (3%)

are expected to select relevant sources of information in order to answer factual questions. Questions in ELI5 are open-ended, typically starting with 'why' and 'how'. Other questions expect a precise answer, such as a date or a figure. Answers in SimpleQA, as its name suggests, can easily be found by a human but they are hard to guess. In contrast, GPQA, the Google-proof query answering dataset, includes questions that are also hard for humans, even with access to the entire Web. GAIA, the general AI assistant dataset, evaluates more general agents but more than half of its questions requires Web search (355/660).

The remaining datasets, listed in Table 3, require action on a Web server. QAWoB, FormWoB and MiniWoB++ are all derived from World of Bits, original work by OpenAI [14]. QAWoB is formulated as a query answering benchmark but unlike above benchmarks, agents must only visit a single Web site (to find a recipe, for example). In FormWoB, agents even have to visite a single Web page. The difficulty in this case is to correctly fill in a complex form (to reserve a flight). MiniWoB++ is derived from MiniWoB, originally published alongside QAWoB and FormWoB. Both variants are a synthetic counterpart to FormWoB, with simpler but more diverse examples. WebShop, Mind2Web and WebArena combine question answering and form completion (to perform a local search, for instance). WebShop exposes simplified pages to agents whereas Mind2Web and WebArena expose raw HTML pages, as found on the Web. Mind2Web did not originally provide screenshots but a later version does (Multimodal-Mind2Web). The latest version of Mind2Web (Online-Mind2Web) is designed for agents that directly access online Web sites rather than offline copies [23]. Evaluating agents online was first introduced for WebVoyager, an agent based on GPT-4V [24].

Broad question answering datasets do not prescribe what Web site to visit to find the right answer but they are distributed with verified sources. Table 3 shows the most frequent sources across datasets. Wikipedia is first on all datasets (SimpleQA, GPQA, GAIA). ELI5 was initially built from various sources but the version that was integrated into the knowledge-intensive language tasks (KILT) benchmark entirely relies on Wikipedia as well [25]. GPQA also relies on data provided by the National Institutes of Health (NIH), in the United States, and LibreTexts, an educational digital common. It is interesting to note that content from these three providers is also available in a more structured way. YAGO, DBpedia and Wikidata all provide RDF representations of Wikipedia pages[3], the NIH PubChem platform serves both HTML and RDF representations[4] and LibreTexts embeds RDF data in HTML pages, as encouraged

[3] see e.g. https://www.wikidata.org/entity/Q132451509
[4] see https://pubchem.ncbi.nlm.nih.gov/docs/rdf

by the Schema.org initiative[5]. In other words, an RDF source exists for 100% of ELI5, 40% of SimpleQA, 37% of GPQA and 18% of GAIA—at least. Yet, to the best of our knowledge, agents evaluated against these datasets ignore structured data.

Agents that are evaluated against these benchmark datasets largely follow the principle seen in the previous section: they rely on tools that provide high-level abstractions. Each task in the GAIA dataset, for instance, is annotated with one or more tool category. Web-related tasks only require a search engine, not navigation. Environments that require action, from World of Bits to WebArena, all send images or text to agents and expect human-like interactions, not HTTP messages. Researchers noted that "many MiniWob++ tasks require clicking or dragging actions that cannot be achieved with DOM-element based action" [26]. Out of the four Web sites used in FormWoB, none actually has a `form` element for the task in its raw HTML representation[6]. In other words, those benchmark datasets carry with them the same limitations as agent frameworks regarding identification, interaction and representation. They hide the links and forms that should drive the action of Web agents, as per the hypermedia-as-the-engine-of-application-state (HATEOAS) principle.

Commercial agents by Anthropic and OpenAI perform well on WebArena and Mind2Web. They respectively complete 61% and 56% of the tasks on Online-Mind2Web, for instance, while the best agent developed by an academic team (remotely accessing GPT-4V) completes 31% of the tasks [23]. In average, seven actions must be performed to complete a single task in this dataset, which clearly indicates that chatbot-based agents understand in some way their environment, originally designed for humans. However, there is no evidence that agents understand the main abstractions of the Web. The fact that agents perform equally well on unrelated benchmarks rather suggests the contrary, in fact. On the general-purpose AgentBench benchmark, agents based on GPT-4 have comparable results on Web-related tasks (taken from WebShop and Mind2Web) and on tasks involving code execution and game simulations [27].

Are there tasks that could only be completed in a Web environment? Original publications about the Semantic Web hint at two core tasks: choosing among Web services and composing Web services. In "Agents and the Semantic Web," Hendler describes a task in which an agent must help a fishing boat caught in a storm [2]. The agent finds several services on the Web, either to download satellite images or to call coast guards for help. In this task, the agent has to choose one of the services, although they are not functionally equivalent. Similar tasks include choosing among several offers for the same product and selecting among contradictory sources of information. In "The Semantic Web," the same year, Berners-Lee, Lassila and Hendler describe a task involving a patient, their family and physicians [28]. The agent must find an appointment with a physician that is compatible with the schedule of all family members. Here, the agent must find physicians in an area and fetch the schedule of several persons which, for privacy reasons, are assumed to be managed by distinct personal information management systems. Each partipant in the task has its own server, for instance a Social Linked Data (Solid) Pod[7], and its own "steward" agent that has privileged access to their data but not others'.

In all those tasks, servers are assumed to be managed by distinct organizations. Information is not and should not be centralized on a single platform. WebArena includes tasks that require cross-site navigation, fulfilling this requirement. For instance, "an agent needs to find out what art museums are located in Pittsburgh by searching Wikipedia. Next, it should identify the location of each museum on a map, optimizing the itinerary based on the information collected. Finally, the agent needs to update the README file in the appropriate repository with the planned route[8]." Cross-site tasks like this one represent only 5.9% of the dataset. Most tasks could have been performed on mobile or desktop applications with a specific user interface. Mind2Web includes no cross-site task.

[5] see e.g. https://search.google.com/test/rich-results/result?id=WlI1Rg7EGr_aZRQcFDfPtg

[6] on jetblue.com and aa.com, a form is added by a script; on alaskaair.com, the page includes a form for mobile browsers, not desktop browsers; on united.com, the page has no form even after execution of all embedded scripts.

[7] see https://solidproject.org/

[8] see https://webarena.dev/

4. Proposal

Some agent frameworks expose Web abstractions to agents (LangGraph and Smolagents have generic tools for HTTP requests, for instance) but most frameworks do not. Some benchmark datasets impose cross-site navigation (WebArena and GPQA have tasks combining information from different Web sites) but on a small set of tasks. More generally, architectural constraints of the Web encourage interoperability but platform-specific tools and tasks, on the contrary, encourage isolation. In light of this opposition, chatbot-based agents should not be considered as Web agents. It is true that the research objective behind the work reviewed in this paper is to build intelligent agents, not specifically Web agents. However, it does not entirely address the original research project initiated by Berners-Lee, Lassila and Hendler. To align with their research objective, this paper introduces criteria to decide whether a task is specific to the Web or not. The criteria, formulated below, are based on several assumptions: to perform a single task, the agent acts as a Web client (it always initiates interactions), it reads and writes resources from one or more Web servers and representations may be served in any format.

Definition 1. *A task performed by an intelligent agent is specific to the Web if all of the following criteria hold:*

1. *the client reads the representation of resources from at least two servers during execution;*
2. *the client reads the representation a resource only if a link to that resource exists in the representation of a previously accessed resource or it is the first resource being read;*
3. *the client writes the representation of a resource only if a form describing the operation exists in the representation of a previously accessed resource;*
4. *the client executes no server-sent code (optional).*

This definition prescribes that all resource representations are in a hypermedia format, exposing links and forms. HTML is the most common hypermedia format but it not the only one. RDF formats (Turtle, JSON-LD and others) are also hypermedia formats, for example. The definition also excludes "intelligent" intermediaries, such as remote chatbot services, because such intermediaries are not interlinked with the rest of the Web. It is also assumed that the set of resources that are read during execution is minimal: no subset is sufficient to complete the task. If an agent always uses a remote search engine to find other resources, criterion 1 is trivially met without this condition.

Question answering tasks in ELI5, SimpleQA, GPQA and GAIA may or not meet criterion 1. For simple questions as in ELI5 and SimpleQA, a single source of information is often enough. GPQA has three levels of difficulty; questions at the highest level (post-graduate level) all require to combine sources and thus meet criterion 1. Criterion 3 does not apply to question answering tasks but the question of whether criteria 2 and 4 are met by GPQA and GAIA is open. WebArena, the only dataset in the other category that meets criterion 1, is also not guaranteed to meet criteria 2 and 4. To answer this research question, agents could rely on RDF representations for navigation—and delegate decision to language models if not available. Unlike RDF crawlers whose purpose is to collect as many RDF resources as possible, such agents would aim to minimize the number of visited resources, following links (exposed as RDF triples) only if they bring closer to a target resource. To find an optimal route connecting art museums in Pittsburgh, for instance, only one hop from Wikipedia to OpenStreetMap is required. Most of the information can first be fetched from Wikidata with the following SPARQL query: `SELECT ?m ?coords WHERE { ?m wdt:P31 wd:Q33506 ; wdt:P131 wd:Q1342 ; wdt:P625 ?coords }`. Then, each result entity has a link to OpenStreetMap, which provides further options to query entities with OpenSearch and find directions via an HTML form.

Wikidata and OpenStreetMap are among the platforms that welcome software agents by adopting technologies similar to RDF to make structured data easily accessible. By doing so, they also acknowledge that the Web is a shared space that encourages coordination. Developing agents that operate at the level of URIs, HTTP and RDF (with raw HTML if needed) is a good way to answer the most difficult queries and achieve the most difficult tasks: queries a single source cannot answer; tasks a single platform cannot perform.

Declaration on Generative AI

The author has not employed any Generative AI tools.

References

[1] J. Hendler, Where Are All the Intelligent Agents?, IEEE Intelligent Systems 22 (2007) 2–3. URL: http://ieeexplore.ieee.org/document/4216971/. doi:10.1109/MIS.2007.62.

[2] J. Hendler, Agents and the Semantic Web, IEEE Intelligent Systems 16 (2001) 30–37. URL: http://ieeexplore.ieee.org/document/920597/. doi:10.1109/5254.920597.

[3] OpenAI, Computer-Using Agent, 2025. URL: https://openai.com/index/computer-using-agent/.

[4] Anthropic, Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku, 2024. URL: https://www.anthropic.com/news/3-5-models-and-computer-use.

[5] Technical Architecture Group, Architecture of the World Wide Web, Volume One, 2004. URL: https://www.w3.org/TR/webarch/.

[6] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, Y. Cao, ReAct: Synergizing Reasoning and Acting in Language Models, in: The Eleventh International Conference on Learning Representations, 2023. URL: https://openreview.net/forum?id=WE_vluYUL-X.

[7] X. Wang, Y. Chen, L. Yuan, Y. Zhang, Y. Li, H. Peng, H. Ji, Executable Code Actions Elicit Better LLM Agents, in: Forty-first International Conference on Machine Learning, 2024. URL: https://openreview.net/forum?id=jJ9BoXAfFa.

[8] B. Zheng, B. Gou, J. Kil, H. Sun, Y. Su, GPT-4V(ision) is a Generalist Web Agent, if Grounded, in: Forty-first International Conference on Machine Learning, 2024. URL: https://openreview.net/forum?id=piecKJ2DlB.

[9] S. Wu, S. Zhao, Q. Huang, K. Huang, M. Yasunaga, K. Cao, V. N. Ioannidis, K. Subbian, J. Leskovec, J. Zou, AvaTaR: Optimizing LLM Agents for Tool Usage via Contrastive Reasoning, in: A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, C. Zhang (Eds.), Advances in Neural Information Processing Systems, volume 37, Curran Associates, Inc., 2024, pp. 25981–26010. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/2db8ce969b000fe0b3fb172490c33ce8-Paper-Conference.pdf.

[10] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, J. Schulman, WebGPT: Browser-assisted question-answering with human feedback, 2022. URL: http://arxiv.org/abs/2112.09332. doi:10.48550/arXiv.2112.09332, arXiv:2112.09332 [cs].

[11] OpenAI, Introducing deep research, 2025. URL: https://openai.com/index/introducing-deep-research/.

[12] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language Models Can Teach Themselves to Use Tools, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems, volume 36, Curran Associates, Inc., 2023, pp. 68539–68551. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

[13] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, d. li, Z. Liu, M. Sun, ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=dHng2O0Jjr.

[14] T. Shi, A. Karpathy, L. Fan, J. Hernandez, P. Liang, World of Bits: An Open-Domain Platform for Web-Based Agents, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 3135–3144. URL: https://proceedings.mlr.press/v70/shi17a.html.

[15] E. Z. Liu, K. Guu, P. Pasupat, P. Liang, Reinforcement Learning on Web Interfaces using Workflow-

Guided Exploration, in: International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=ryTp3f-0-.

[16] S. Yao, H. Chen, J. Yang, K. Narasimhan, WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 20744–20757. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

[17] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, Y. Su, Mind2Web: Towards a Generalist Agent for the Web, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems, volume 36, Curran Associates, Inc., 2023, pp. 28091–28114. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/5950bf290a1570ea401bf98882128160-Paper-Datasets_and_Benchmarks.pdf.

[18] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, G. Neubig, WebArena: A Realistic Web Environment for Building Autonomous Agents, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=oKn9c6ytLx.

[19] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, M. Auli, ELI5: Long Form Question Answering, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 3558–3567. URL: https://aclanthology.org/P19-1346/. doi:10.18653/v1/P19-1346.

[20] J. Wei, K. Nguyen, H. W. Chung, J. Jiao, S. Papay, M. Glaese, J. Schulman, L. Fedus, Introducing SimpleQA, 2024. URL: https://github.com/openai/simple-evals.

[21] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, S. R. Bowman, GPQA: A Graduate-Level Google-Proof Q&A Benchmark, in: First Conference on Language Modeling, 2024. URL: https://openreview.net/forum?id=Ti67584b98.

[22] G. Mialon, C. Fourrier, T. Wolf, Y. LeCun, T. Scialom, GAIA: a benchmark for General AI Assistants, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=fibxvahvs3.

[23] T. Xue, W. Qi, T. Shi, C. H. Song, B. Gou, D. Song, H. Sun, Y. Su, An Illusion of Progress? Assessing the Current State of Web Agents, 2025. URL: http://arxiv.org/abs/2504.01382. doi:10.48550/arXiv.2504.01382, arXiv:2504.01382 [cs].

[24] H. He, W. Yao, K. Ma, W. Yu, Y. Dai, H. Zhang, Z. Lan, D. Yu, WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 6864–6890. URL: https://aclanthology.org/2024.acl-long.371/. doi:10.18653/v1/2024.acl-long.371.

[25] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, V. Plachouras, T. Rocktäschel, S. Riedel, KILT: a Benchmark for Knowledge Intensive Language Tasks, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 2523–2544. URL: https://aclanthology.org/2021.naacl-main.200. doi:10.18653/v1/2021.naacl-main.200.

[26] P. C. Humphreys, D. Raposo, T. Pohlen, G. Thornton, R. Chhaparia, A. Muldal, J. Abramson, P. Georgiev, A. Santoro, T. Lillicrap, A data-driven approach for learning to control computers, in: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato (Eds.), Proceedings of the 39th International Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 9466–9482. URL: https://proceedings.mlr.press/v162/humphreys22a.html.

[27] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, J. Tang, AgentBench: Evaluating LLMs as Agents, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=zAdUB0aCTQ.

[28] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American (2001) 36–43. URL:

https://lassila.org/publications/2001/SciAm.pdf.