OpenWebAgent: An Open Toolkit to Enable Web Agents on Large Language Models

Iat Long Iong[1*†] **Xiao Liu**[1*] **Yuxuan Chen**[1†] **Hanyu Lai**[1†] **Shuntian Yao**[2†]
Pengbo Shen[3†] **Hao Yu**[1†] **Yuxiao Dong**[1‡] **Jie Tang**[1‡]
[1]Tsinghua University [2]Beijing University of Posts and Telecommunications
[3]University of the Chinese Academy of Sciences
rongyl20@mails.tsinghua.edu.cn, shawliu9@gmail.com

Abstract

We introduce OpenWebAgent, an open toolkit designed to optimize web automation by integrating both large language models (LLMs) and large multimodal models (LMMs). This toolkit focuses on enhancing human-computer interactions on the web, simplifying complex tasks through an advanced HTML parser, a rapid action generation module, and an intuitive user interface. At the core of OpenWebAgent is an innovative web agent framework that uses a modular design to allow developers to seamlessly integrate a variety of models and tools to process web information and automate tasks on the web. This enables the development of powerful, task-oriented web agents, significantly enhancing user experience and operational efficiency on the web. The OpenWebAgent framework, Chrome plugin, and demo video are available at https://github.com/THUDM/OpenWebAgent/.

1 Introduction

As the Internet becomes an integral part of everyday life, the complexity of tasks that users want to automate on web platforms continues to grow (Van der Aalst et al., 2018). Modern web users expect interfaces that are not only intuitive and visually appealing but also capable of intelligent (Davenport and Kirby, 2016), predictive interactions that streamline complex tasks.

Traditional web automation tools, such as Robotic Process Automation (RPA) tools, have been instrumental in reducing manual effort (Syed et al., 2020), but fall short in areas such as contextual understanding, flexibility (Hallikainen et al., 2018), and user accessibility. These tools often require complex setups and significant technical expertise, limiting their usefulness to a narrow audience. In addition, the lack of a unified system architecture among existing tools poses significant challenges for developers seeking to integrate or innovate on top of these platforms, hindering the advancement of web automation technologies.

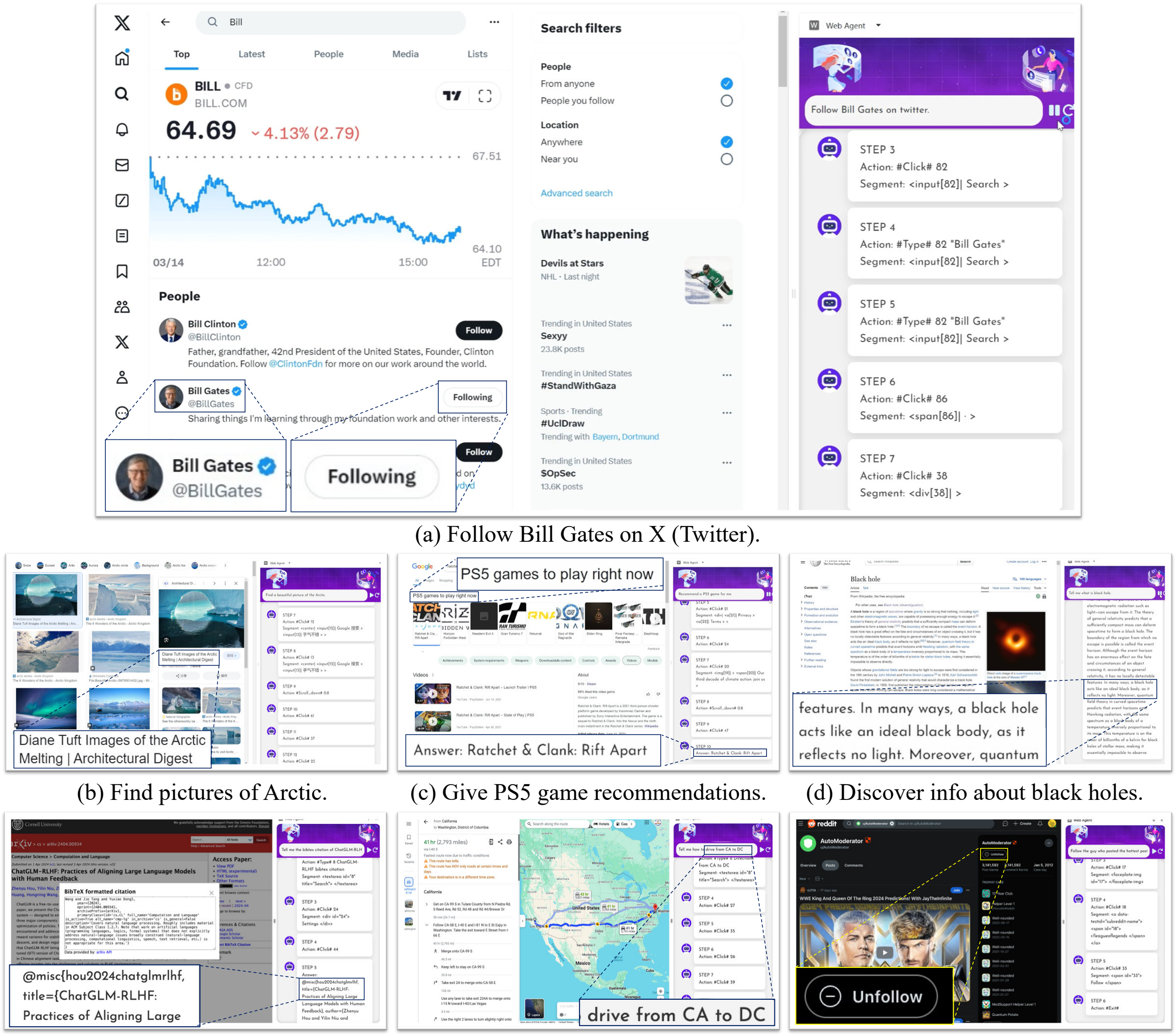
Nevertheless, the field has advanced significantly with deep learning technologies, especially after Google introduced the Transformer model (Vaswani et al., 2017). The capabilities of large-scale pre-trained models, like OpenAI's GPT series (Achiam et al., 2023; OpenAI, 2024a), have improved text generation, semantic understanding, and logical reasoning (Brown et al., 2020; Chan et al., 2022). This has made web automation tools using LLMs and LMMs more feasible. Recent work such as AutoWebGLM (Lai et al., 2024), shows LLMs can handle various web tasks but the lack of multimodal inputs limits their capabilities, highlighting the need for multimodal web agents.

Presented System. We present OpenWebAgent, a toolkit for advanced web interactions with innovative modules that allow developers to integrate any language or multimodal model for web automation. Figure 1 shows the task execution results of OpenWebAgent System with GPT-4 (Achiam et al., 2023) and AutoWebGLM (Lai et al., 2024) as the action generation models. OpenWebAgent includes an interactive web plugin and a modularly designed server, allowing it to execute tasks directly and autonomously on webpages while providing real-time feedback. It stands out for its efficiency and user accessibility compared to other web automation tools, thanks to these features:

- **High-Performance HTML Parser.** This parser optimizes performance by simplifying complex HTML into a more straightforward format (§3.1), enabling OpenWebAgent to process web content with enhanced accuracy and speed. It reduces HTML length by 99% and the number of elements by 97% (§5.1), ensuring efficient operation on any website. (§5.2)

[*]Equal contribution.
[†]Work done while these authors interned at Zhipu AI.
[‡]Corresponding Authors: YD and JT.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 72–81
August 11-16, 2024



(a) Follow Bill Gates on X (Twitter).

(b) Find pictures of Arctic.

(c) Give PS5 game recommendations.

(d) Discover info about black holes.

(e) Find the BibTeX of a paper.

(f) Get Driving Directions.

(g) Follow the qualifying account.

Figure 1: Examples of OpenWebAgent performing daily tasks. In these examples, GPT-4 is employed as the action generation model in (a) – (d), while AutoWebGLM is used in (e) – (g).

- **Modular System Design.** OpenWebAgent integrates multimodal inputs such as action history, parsed HTML and screenshots for LLMs and LMMs to create coherent action plans that match user intent. Users can modify, pause, or reset tasks at any time (§4.2), offering flexibility and ease of use. The modular design (§3.2) allows easy model integration and module replacement by developers.
- **Streamlined User Interface.** The plugin requires no complex setup and is ready to use immediately after download. Its simple and attractive interface (§4.1) lets users track the process and sequence of operations easily, with task execution controlled by a few simple buttons, ensuring efficiency and ease of use.

These innovations place OpenWebAgent at the forefront of web automation technology. Through its advanced capabilities, OpenWebAgent is not just a tool but a paradigm shift in how humans interact with and harness the power of the web for automated tasks.

Contributions. (1) We implement a powerful HTML parser engine that simplifies complex webpages into a more accessible format. (2) We design a ready-to-use web plugin automation tool that enables users to perform any desired action on any webpage. (3) We create a versatile web agent framework, allowing developers to easily integrate any LLM or LMM for web automation. This framework offers unprecedented ease in developing robust, task-oriented web agents. In summary, OpenWebAgent contributes both technically and conceptually to the understanding of the boundaries of human-machine collaboration and giving an attempt to develop the future of web automation.

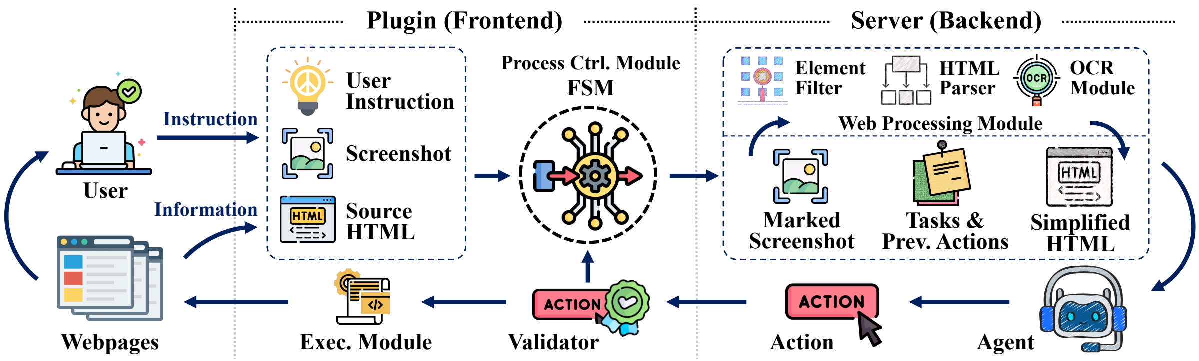


Figure 2: System Design. Our system has two main components: the frontend plugin and the backend server. The frontend plugin collects page information, performs webpage actions, and controls operations using a finite state machine (FSM). The backend server processes data and organizes prompts for the agent to predict actions.

2 Related Work

Developing an efficient toolkit for web browsing agents is challenging, especially in integrating decision language models with diverse modules for processing webpage information. This section provides an overview of related research.

Language Models (LMs). Since Google proposed Transformer (Vaswani et al., 2017), Large Language Models (LLMs) have evolved rapidly. Notable models include OpenAI's GPT-4 (Achiam et al., 2023), Google's Gemini (Google, 2023), PaLM-2 (Chowdhery et al., 2023), and Gopher (Rae et al., 2021), Anthropic's Claude-2 (Anthropic, 2023), Meta's LLaMA series (Touvron et al., 2023a,b) and OPT (Zhang et al., 2022), as well as Mistral's Mixture of Experts models (Jiang et al., 2024). Other notable contributions include GLM-130B (Zeng et al., 2022) and BLOOM (Scao et al., 2022). These models, pre-trained on vast datasets, excel in various NLP tasks.

Large Multimodal Models (LMMs) have become the primary focus of research to address a wider range of tasks. OpenAI led the way by launching high-performance models such as GPT-4-Turbo (OpenAI, 2024b) and GPT-4o (OpenAI, 2024a). The open-source community has also introduced multimodal models like LLaVA (Liu et al., 2023a), CogVLM (Wang et al., 2023), and Qwen-VL (Bai et al., 2023). These LMMs have inspired new approaches in Agent research.

Smaller, cost-effective models are preferred due to the high deployment costs of large models, with users prioritizing task-specific effectiveness. Open-source projects like LLaMA3-8B (Meta, 2024), Vicuna-7B (Chiang et al., 2023), and ChatGLM4-9B (GLM et al., 2024) show comparable capabilities to larger models in some areas.

Web Automation Systems. Previous projects, such as WebGPT (Nakano et al., 2021) and WebGLM (Liu et al., 2023b), have effectively integrated language models with web environments, mainly for question-answering tasks using internet data. These models are excellent at information retrieval for QA (Rajpurkar et al., 2016; Nguyen et al., 2016; Berant et al., 2013; Kwiatkowski et al., 2019), but they cannot perform complex or interactive web-based tasks.

Recent projects like AutoGPT[1] use multiple ChatGPT agents for self-prompting and web operations through a plan-execute-reflect cycle. The GPT-4V-ACT framework[2] uses the Set-of-Mark method (Yang et al., 2023) to mark screenshots and then employs GPT-4V (OpenAI, 2024b) to generate operations, but it struggles with real web pages due to insufficient operation instructions. AutoWebGLM (Lai et al., 2024) is based on the fine-tuned ChatGLM3-6B (GLM et al., 2024) model. However, it lacks image input data, limiting its performance in real-world web scenarios.

Other initiatives such as MindAct (Deng et al., 2023) involve extensive interactions to select webpage elements, suggesting a need for more efficient processes. CC-Net (Mishra et al., 2019) leverages a vast visual data set and learning techniques to manipulate web components effectively. Conversely, CogAgent (Hong et al., 2023) focuses on using visual input to generate web operation methods, while WebAgent (Gur et al., 2024) utilizes HTML-T5 and the large-scale Flan-U-Plam model (Chung et al., 2022) to control webpages, though the model's size limits its deployment.

[1] https://github.com/Significant-Gravitas
[2] https://github.com/ddupont808/GPT-4V-Act

3 The OpenWebAgent System

OpenWebAgent is rooted in principles of intuitive design, flexibility, and comprehensive functionality, aiming to provide a user-centric approach to web automation. The system is inherently adaptable, and designed to allow users to easily customize it for a range of complex automation tasks. Its main objective is to develop a toolkit that is more responsive and goes beyond the limitations of traditional web agents. This toolkit does more than execute user tasks with simple pre-defined commands; it is engineered to fully analyze, decompose, and process each task to ensure thorough and efficient automation.

3.1 HTML Parsing Techniques

The content of HTML webpages is intricate and complex. Therefore, it should be effectively simplified before being fed into the parsing model.

Simplification aims to distill the most important information while eliminating excessive or disruptive elements that could make it difficult for the model to understand. It is crucial to maintain the basic structure of HTML and its essential content information during this process. This ensures that the model can understand and use these details for efficient webpage parsing.

Using algorithm 1 can effectively transform the element tree into a more concise representation. We can judge whether an element should be retained by determining the clickability of the element, noting that nodes near the retained element are generally able to provide more useful information and therefore have a higher retention value. Therefore, we can adopt a recursive approach to obtain the ancestor nodes, child nodes and sibling nodes of the retained element part. Finally, pruning can be done according to the information content starting from the leaf nodes.

With the processing algorithm described above, the complex HTML can be simplified into a format that is easier for the model to interpret and manipulate, thus improving the model's performance in web parsing tasks.

3.2 Interaction Workflow

OpenWebAgent's design philosophy and objectives aim to achieve a harmonious balance between advanced technological capabilities and user-friendly interaction, redefining standards for human-computer interaction in web automation. To improve the flexibility and usability of our toolkit, we modularize it into several key components as shown in Figure 2. The network processing module and the action generation module are deployed as unified services in the backend. Meanwhile, the process control module and the execution module are integrated into the plugin.

Algorithm 1: HTML Simplifier

```
Data: dom tree tree, neighbor coefficient n
Result: pruned tree tree, kept elements kp
nodes, kp ← set(), list()
for e in tree.element do    // selector
    if not (onTop(e) and onScreen(e))
      then continue
    if isClickableTag(e.tagname) or
      haveJSaction(e.attrib) or
      e.cursor = pointer or
      e.classes.include(button) then
        kp.push(e)
        nodes.push(e)
        nodes.push(getNbr(e, n))
    end
end
for e in reversed(tree) do    // pruner
    if not e in nodes or not (e has text or
      attrib or e is root or
      len(e.children) > 1) then
        tree.remove(e)
    end
end
```

Web Processing Module. This module extracts useful elements of HTML, simplifies HTML input, performs OCR on screenshots, and adds element labels to screenshots. See §3.1 for details of the methods and processes.

Action Generation Module. The main purpose of this module is to predict the next action based on the user's task and the current webpage context. At this stage, we provide a prompt for the LLM that includes the current task, the simplified HTML of the webpage, and previous command sequences, and for the LMM, we also provide labeled screenshots for the model to use. The model outputs the next action in natural language, which we match against a pre-defined action space, and returns the action name and parameters if the match is successful.

In this module, models are accessed through interfaces, which means that developers creating new web agents can effortlessly integrate any model into our toolkit by simply setting up an API interface for accessing the model. It is important to note that we have reserved an interface to a visual processing module in the toolkit, located between the web processing module and the action generation module, to serve better the needs of developers working with multimodal agents.

To address the lack of image information in the language-based agent, the module is pre-configured with an OCR interface that takes a screenshot of a webpage as input and returns the webpage information contained in the screenshot. LMM agent developers have the option to replace the preconfigured modules and integrate their own vision modules into our toolkit, which simplifies the development of multimodal web agents.

Execution Module. This module is used to execute specific action instructions on a webpage. When the module receives an action instruction from the action generator module, it looks for the element that actually needs to be operated on and then executes the action on the webpage using a predefined script. When the action is completed, it provides a response feedback to ensure that the model is aware of the execution of the action to adjust the plan.

Process Control Module. As shown in Figure 3, this module is implemented using a finite state machine. The main purpose of this is to coordinate the execution of tasks and to facilitate the transfer of information between the above modules.

This module serves as the primary interface for user interaction. It receives various inputs, such as user task commands and control commands (e.g. start, pause, reset). The module also records user input tasks and previous action history. When the user issues a start command, the module first fetches the HTML source and screenshot of the current webpage, and information about the clickable elements, and passes them to the web processing module (including the visual processing module). Additionally, the module sends the task and action history to the action generation module.

The module sends task and action history to the action generation module and waits for a response from the action generation module. It then parses the action into various parameters. If a valid web action returns, its details are sent to the execution module. Once a response arrives from the execution module, the action history updates and the process of retrieving webpage information repeats.

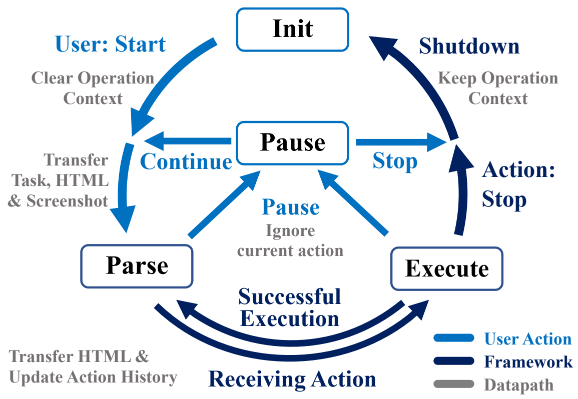


Figure 3: FSM Design.

When the process is complete, the user is notified by the module.

This workflow enables real-time user interaction, users can send control commands such as pause, reset, or update their task description at any time. The process control module adapts accordingly, ensuring flexible and efficient interaction.

4 Demonstration

4.1 System Interface

The plugin interface is simple and easy to use, as shown in Figure 4. It consists of three main parts:

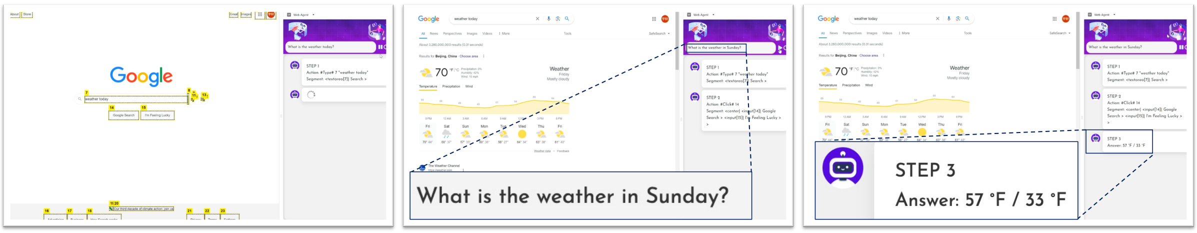
- **Input Box**: For entering tasks to be executed.
- **Control Buttons**: For managing task execution, starting with "▷ run" and "↻ reset" . During execution, "॥ pause" replaces "▷ run" , allowing the user to control the process.
- **Feedback Panel**: Shows the executed actions and the model's responses.

Figure 4: The system interface of OpenWebAgent.

4.2 Usage Example

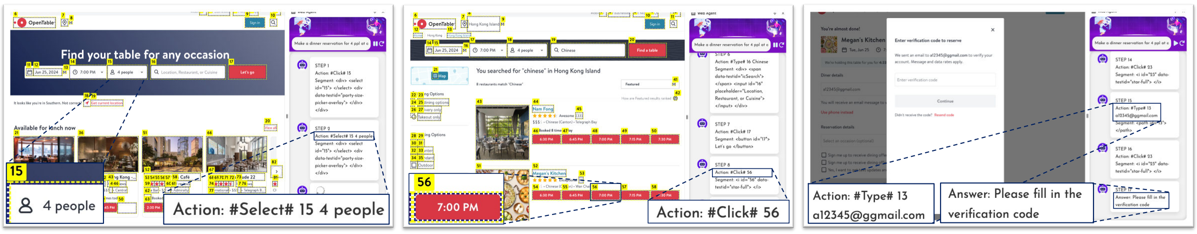
As shown in Figure 5, we illustrate the interaction process and execution results of our toolkit by integrating GPT-4 into our toolkit and executing a sample task *"What is the weather like today?"*.

Task Execution. The task begins with Google, a standard browser homepage that does not provide weather information. First, we can put the query *"What is the weather like today?"* into the task box and click the "▷ run" button to initiate the task. A



(a) Start of execution. (b) Change current task. (c) Get requested information.

Figure 5: Execution flow of OpenWebAgent. Initially (a) execute *"What is the weather like today?"*, then at (b) modify the task to *"What is the weather in (on) Sunday?"*, and finally (c) get the answer for the task.



(a) Set the number of diners. (b) Select restaurant and dining time. (c) Ask user to enter verification code.

Figure 6: Example of using OpenWebAgent with AutoWebGLM. The task is *"Make a dinner reservation for 4 at a Chinese restaurant with the email a12345@ggmail.com."*. This example demonstrates that OpenWebAgent can handle various types of elements.

loading icon will appear on the feedback box to indicate that our plugin has initiated the retrieval of information from the webpage. After a brief interval, the plugin generates the instruction, "`#Type# 5 weather today`". The webpage displays that *"weather today"* has been entered into the input box, thereby suggesting that the action has been executed successfully.

Task Management. After several actions on the webpage, the user is presented with a weather forecast, as depicted in Figure 5(b). At this juncture, the user has the option to pause the task by clicking the "⏸ `pause`" button. They can then update the task instructions to *"What is the weather on Sunday?"* before resuming execution by clicking "▷ `run`" (which serves as "`continue`" here). The plugin will adapt to the modified user task and modify the execution flow accordingly.

Task Completion. Following a series of actions on the webpage, the LLM (GPT-4) can complete the user's task based on the information available. Our plugin then responds to the user's query by returning a message, "`Answer: 57°F / 33°F`", in the feedback box and completes the process.

Figure 6 shows how OpenWebAgent with AutoWebGLM handles a restaurant reservation task. This demonstrates that our framework provides the necessary information and actions to support the model in performing complex tasks. In addition, as shown in Figure 1, our plugin can also perform various web tasks, such as shopping, socializing and information seeking to satisfy users' diverse web browsing needs. This proves that our plugin has the following characteristics:

- **Flexibility**: Users can use our plugin to accomplish various tasks on any webpage, in any state, anytime, anywhere.
- **Efficiency**: Each module in our plugin is optimized for performance, and the time taken for each step of the action depends largely on the time taken to call LLM. Therefore, our plugin executes extremely efficiently.
- **Robust Interactivity**: Our plugin receives user interaction at any moment during execution. Users can receive feedback and take control in real time.

5 Evaluation

5.1 HTML Parser Performance

Experimental setup. We selected six categories of frequently visited websites from Similarweb[3] and randomly tested the effectiveness of the parser by selecting dozens of pages from each website, and the results are shown in Table 1.

[3] https://www.similarweb.com/top-websites

Website	Length			Elements			Time
	# before	# after	reduction (%)	# before	# after	reduction (%)	msec
E-commerce	725,948	2,174.5	99.62	2,580.2	56.34	96.38	5.52
- Amazon	1,103,437	2,643.3	99.71	4,329.8	59.87	97.99	8.30
- eBay	679,852	2,108.7	99.65	2,138.3	46.28	97.70	5.11
- Taobao	394,555	1,771.4	99.51	1,272.6	62.86	93.50	3.14
Entertainment	825,534	2,069.1	99.48	2,705.0	39.33	98.00	5.42
- Bilibili	1,669,682	2,217.6	99.62	3,144.2	53.00	97.12	7.19
- Spotify	317,223	1,947.6	99.39	1,756.4	23.14	98.60	3.75
- Fandom	489,698	2,042.3	99.43	3,217.3	41.85	98.29	5.32
Forum	762,945	2,846.3	99.61	2,983.3	50.88	97.98	5.68
- Reddit	872,041	3,258.9	99.63	2,838.8	47.25	98.00	5.60
- Quora	653,849	2,433.7	99.59	3,127.7	54.50	97.96	5.75
Knowledge	452,532	4,584.8	98.71	2,630.8	75.11	96.62	4.38
- Wikipedia	440,264	6,135.1	98.37	2,479.0	95.33	96.10	4.33
- Baidu-baike	464,801	3,034.5	99.05	2,782.6	54.89	97.15	4.42
News	763,077	3,731.0	98.58	1,821.4	50.91	96.49	4.53
- Yahoo	1,829,666	4,959.4	99.53	2,843.6	40.22	98.47	8.18
- Yahoo-JP	317,049	2,457.3	99.23	1,693.8	74.87	95.48	3.12
- QQ	142,515	3,776.4	97.01	926.7	37.85	95.54	2.27
Social Media	1,679,296	1,547.1	99.81	3,389.6	47.37	97.95	8.96
- Facebook	3,332,936	1,663.8	99.94	6,417.8	47.67	99.13	14.94
- Instrgram	1,176,319	768.6	99.93	1,172.2	25.43	97.77	6.76
- X	528,634	2,208.7	99.57	2,578.7	69.00	96.96	5.18
Overall	900,782	2,714.2	99.32	2,669.9	52.12	97.24	5.83

Table 1: HTML simplification results on various sites.

Results Analysis. The HTML simplifier effectively reduces the complexity of web pages across various websites. It significantly reduces the number of actionable elements and the length of HTML text, with simplification rates exceeding 97% and 99% respectively. The tool operates quickly, even in dense environments like Facebook, with an average processing time of just 5.83 milliseconds. This rapid performance demonstrates the tool's practicality for real-world applications, enabling quicker and more focused web interactions by emphasizing essential content.

5.2 Efficiency

Experimental setup. Following the HTML Parser Performance testing methodology, we selected 12 websites from SimilarWeb. The system was assigned 80 web navigation tasks across these diverse websites. Throughout these tasks, we meticulously recorded the response times of different components. The outcomes of this evaluation are presented in Table 2.

Results Analysis. The results indicate that network transmission time makes up 70% of the system's operational time, primarily due to connections to remote servers. Additionally, the model's prediction activities, particularly with the GPT-4-turbo model, account for 23% of the runtime. To improve efficiency, future enhancements should focus on optimizing web page transmission, such as by locally simplifying web pages to reduce data volume by 99%, thus saving time and enhancing performance.

	Fetch	Parse	Predict	Network	Execute
Time (ms)	510.3	71.2	2,405.7	7,166.4	31.2
Percentage	5.0	0.7	23.6	70.3	0.3

Table 2: System Efficiency.

6 Conclusion

OpenWebAgent represents a paradigm shift in web-based human-computer interaction. It promises to improve user experience and productivity by automating a variety of web tasks efficiently and intuitively. It provides a convenient framework for the development of web agents based on large language models (LLMs) and large multimodal models (LMMs) through advanced HTML parsing capabilities, a modularly designed system, a friendly user interface, and the visualization of the task execution process.

Limitations

While OpenWebAgent boasts remarkable capabilities, it also has its limitations:

- Its performance may falter on complex or unconventional webpages, as it depends on understanding web structures.
- The tool is intended for general purposes and might not perform optimally for tasks that require specialized knowledge.
- The capacity of OpenWebAgent to execute web page operations is substantially influenced by the capabilities of the underlying model, including the ability to comprehend web page elements and perform image recognition.
- Although OpenWebAgent is currently efficient, its backend design needs to be improved to meet the needs of large-scale applications and faster web operations response.

Future developments will address these limitations and improve its applicability and performance.

Ethics Statement

Intended Use. OpenWebAgent is designed to assist users in automating web browsing tasks.

Potential Misuse. OpenWebAgent can perform tasks on the World Wide Web, however, we cannot prevent users from using this tool for network attacks, such as social forum spamming, DDoS attacks, or unauthorized data extraction.

Acknowledgements

This work is supported by Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2022ZD0118600, Natural Science Foundation of China (NSFC) 62276148 and 62425601, the New Cornerstone Science Foundation through the XPLORER PRIZE.

References

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

Anthropic. 2023. Model card and evaluations for claude models. Technical report, Anthropic.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Stephanie CY Chan, Ishita Dasgupta, Junkyung Kim, Dharshan Kumaran, Andrew K Lampinen, and Felix Hill. 2022. Transformers generalize differently from information stored in context vs in weights. *NeurIPS MemARI Workshop*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Thomas H Davenport and Julia Kirby. 2016. *Only humans need apply: Winners and losers in the age of smart machines*. Harper Business New York.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pages 28091–28114.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools.

Gemini Team Google. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A real-world webagent with planning, long context understanding, and program synthesis. In *The Twelfth International Conference on Learning Representations*.

Petri Hallikainen, Riitta Bekkhus, and Shan L Pan. 2018. How opuscapita used internal rpa capabilities to offer services to clients. *MIS Quarterly Executive*, 17(1).

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2023. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Yu Hao, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. AutowebGLM: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. In *Advances in neural information processing systems*, volume 36, pages 34892–34916.

Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023b. Webglm: Towards an efficient web-enhanced question answering system with human preferences. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 4549–4560, New York, NY, USA. Association for Computing Machinery.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. Technical report, Meta.

Suraj Mishra, Peixian Liang, Adam Czajka, Danny Z Chen, and X Sharon Hu. 2019. Cc-net: Image complexity guided network compression for biomedical image segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 57–60. IEEE.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *choice*, 2640:660.

OpenAI. 2024a. Hello gpt-4o. Technical report, OpenAI.

OpenAI. 2024b. New models and developer products announced at devday. Technical report, OpenAI.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Rehan Syed, Suriadi Suriadi, Michael Adams, Wasana Bandara, Sander JJ Leemans, Chun Ouyang, Arthur HM ter Hofstede, Inge van de Weerd, Moe Thandar Wynn, and Hajo A Reijers. 2020. Robotic process automation: contemporary themes and challenges. *Computers in Industry*, 115:103162.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wil MP Van der Aalst, Martin Bichler, and Armin Heinzl. 2018. Robotic process automation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. 2023. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*.

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

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.