Detecting Pipeline Failures through Fine-Grained Analysis of Web Agents

Anonymous ACL submission

Abstract

Web agents powered by large language models (LLMs) can autonomously perform complex, multistep tasks in dynamic web environments. However, current evaluations mostly focus on the overall success while overlooking intermediate errors. This limits insight into failure modes and hinders systematic improvement. This work analyzes existing benchmarks and highlights the lack of fine-grained diagnostic tools. To address this gap, we propose a modular evaluation framework that decomposes agent pipelines into interpretable stages for detailed error analysis. Using the SeeAct framework and the Mind2Web dataset as a case study, we show how this approach reveals actionable weaknesses missed by standard metrics - paving the way for more robust and generalizable web agents.

1 Introduction

AI agents powered by large language models (LLMs) are increasingly deployed in real-world applications that require complex, multistep decision-making, such as coding assistance (Qiao et al., 2023), question answering (Liu et al., 2023), automated fact verification (Sun et al., 2023; Xiong et al., 2025) and web navigation (Yin et al., 2024; He et al., 2024a; Zheng et al., 2024, 2025). These systems typically decompose tasks into modular pipelines, allowing structured reasoning across several intermediate steps. However, evaluation methods predominantly focus on end-to-end task success, offering limited visibility into intermediate reasoning and decision-making processes. This coarse-grained perspective obscures how and why agents fail - hindering systematic debugging, error diagnosis, and safe deployment in real-world scenarios where failures can lead to inefficiencies, degraded user experiences, or unintended behavior.

Web navigation presents a particularly challenging setting for LLM-based agents, requiring multimodal reasoning over structured HTML, natural language, and visual elements. In this context, agents must interpret dynamic and often ambiguous information while executing a sequence of interdependent actions. Small errors in early stages - such as misinterpreting context or selecting the wrong subgoal - can propagate through the pipeline and lead to final task failure. Yet, most existing benchmarks and studies, such as those based on the Mind2Web dataset (Zheng et al., 2024; Deng et al., 2024), evaluate only final task outcomes. This limits our understanding of where errors occur and how agent decisions break down in practice.

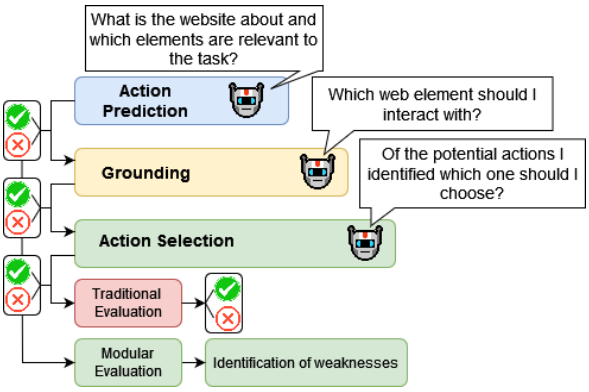

Figure 1: Modular evaluation of the SeeAct agent reveals performance across distinct pipeline stages.

To address this gap, we propose a *modular evaluation* that decomposes web agent pipelines into interpretable stages, including subgoal planning, information grounding, and action selection, and enables fine-grained error analysis across each step. Our goal is not only to expose failure modes more systematically, but also to motivate a shift toward diagnostic evaluation practices for LLM-based agents.

As a case study, we apply our modular evaluation to SeeAct (Zheng et al., 2024), a multimodal web agent that uses vision-language models (VLMs) to perceive and act within web interfaces. Our

contributions are as follows:

- Introduction of a **a conceptual modular evaluation** that captures reasoning quality at each stage of the agent pipeline, providing a comprehensive error analysis.
- **Extending of the SeeAct architecture** by enhancing input representations and improving the heuristic action selection.
- **Augmentation of the Mind2Web evaluation protocol** by introducing alternative valid action annotations, addressing limitations of rigid single-ground-truth assumptions and better reflecting real-world flexibility.

Our experiments show that modular evaluation reveals systematic challenges such as context fragmentation, grounding errors, and ambiguity in HTML-based interfaces - issues often missed by standard metrics. By enabling step-wise diagnosis, our approach supports debugging, robustness testing, and system improvement - key capabilities for advancing reliable and generalizable LLM-based web agents. We release[1] our evaluation toolkit and SeeAct reimplementation to facilitate further research in this direction.

2 Related Work & Background

2.1 Web Agents

Web agents are systems designed to execute action sequences on web interfaces based on natural language instructions (Mazumder and Riva, 2021; Xu et al., 2021). Traditional agents have struggled with diverse layouts and the complexity of web structures. Recent advances in large language models (LLMs) have enabled agents to infer contextually appropriate actions from natural language inputs, greatly expanding their versatility across tasks (Furuta et al., 2024; Gur et al., 2023; Sodhi et al., 2024; Wang et al., 2024).

A persistent challenge lies in effectively interpreting HTML content, which often lacks semantic clarity or task-specific grounding, making accurate action selection difficult (Deng et al., 2024; Gur et al., 2024; Kim et al., 2024; Lo et al., 2023). To address this, several systems incorporate vision-language models (VLMs) that combine visual perception with language understanding—improving both generalization and task success rates. Zheng et al. (2024) introduced a multimodal agent that uses screenshots of web pages for visual understanding and acting on the web. Hong et al. (2024) extended this idea with a generalist agent capable of reasoning across diverse visual domains. Shahbandeh et al. (2024) further integrate multimodal inputs to enhance contextual understanding in their functionality-guided navigation agent. Several other systems have also leveraged VLMs to boost generalization (He et al., 2024b; Zhang et al., 2025)

Web Agent	Planning & Grounding	Evaluation Granularity
He et al. (2024b)	✗	End-to-End
Shahbandeh et al. (2024)	✓	End-to-End
Lai et al. (2024)	✓	End-to-End
Iong et al. (2024)	✓	End-to-End
Zheng et al. (2024)	✓	End-to-End
Ours	✓	Fine-Grained

Table 1: Comparison of evaluation granularity.

2.2 Evaluation of Web Agents

Despite the increasing complexity of web agents, most evaluation protocols remain coarse-grained, focusing solely on the final, end-to-end task success, instead of taking also intermediate steps into account. Zhou et al. (2023), for instance, evaluate agents based on final task correctness, and Li and Waldo (2024) introduce a taxonomy of web actions to categorize failure types. Other approaches such as Pan et al. (2024), Mühlbacher et al. (2024) and Xu et al. (2025) propose multi-level or composite metrics (e.g., *Success*, *Partial Success*), yet they still operate at the task level and do not expose failure propagation across intermediate steps.

These evaluations provide only limited insight into why agents fail, particularly in multistep settings in which reasoning, grounding and execution are interacting closely with each other. Without visibility into intermediate stages, debugging becomes difficult, and iterative improvement remains guesswork. In contrast, our work introduces a modular evaluation framework that decomposes the agent pipeline into interpretable stages, enabling fine-grained performance tracking at the level of planning and grounding. This not only supports targeted debugging but also opens the door to better understanding of failure sources and their frequency across tasks.

Table 1 compares recent web agents along key dimensions relevant to evaluation: whether they implement explicit planning and grounding, whether

[1] Anonymous Github Repo

they support modular evaluation, and the granularity of their evaluation protocols. As shown, all prior work uses end-to-end evaluation, with no mechanism for inspecting or isolating failures within the pipeline. Our improved SeeAct agent is the only system to support modular evaluation with stage-level granularity, while also achieving a higher success rate than the original SeeAct baseline. This comparison highlights the broader need for diagnostic tools that go beyond task-level metrics to enable deeper analysis of web agent behavior.

3 Methodology

We propose a modular evaluation for LLM-based web agents that enables fine-grained diagnosis of reasoning and grounding failures. Rather than assessing only the final task outcome, our approach decomposes an agent pipeline into interpretable stages and evaluates each step independently. This helps to localize error sources, understand failure propagation, and guide system improvement.

3.1 A Modular Evaluation Framework

Our framework evaluates web agents by decomposing their complex behavior into distinct, interpretable stages (Figure 1). This modular approach allows for fine-grained diagnosis, as small errors in early stages can otherwise cascade and lead to task failure in ways that are obscured by end-to-end metrics. Each stage is evaluated with tailored metrics designed to reflect reasoning quality and alignment with the ground-truth action.

Stage 1: Action Prediction (Planning) In this initial stage, the agent observes its environment (e.g., HTML, screenshots) and, based on the task instruction, performs reasoning to identify relevant elements and decide on the next abstract action. We assess this stage with two metrics:

- **Relevant Element Accuracy (RE Acc.):** Measures the effectiveness of candidate generation. It is the percentage of instances where the ground-truth element is successfully included within the set of candidate elements presented to the agent.
- **Action Prediction Accuracy (AP Acc.):** Measures the core planning decision. Given a candidate set known to contain the ground-truth element, this is the percentage of times the agent's abstract prediction correctly identifies that element as the target for its next action.

Stage 2: Grounding. Here, the agent's abstract intent from the planning stage is translated into a concrete, executable action triplet (*Element, Action, Value*). We evaluate this with a single, strict metric:

- **Grounding Accuracy:** The percentage of instances where the fully-grounded action triplet produced by this stage exactly matches the ground-truth triplet. For evaluations involving parallel batches, we only assess the batch that contains the ground-truth action in its candidate set.

Stage 3: Action Selection. This final stage resolves the outputs from the agents that have batch processing. Since each webpage section is processed as an independent batch, multiple viable, grounded actions are often generated. The task is to select the single correct action from this aggregated set of candidates. We assess two distinct strategies:

- **First Viable:** A simple heuristic baseline that measures the accuracy if we simply choose the first valid (non-None) candidate from the aggregated list.
- **LLM Select:** Our proposed strategy measures the accuracy of an LLM prompted to evaluate all available viable candidates and choose the single best one.

This modular structure supports plug-in evaluation on any agent pipeline that outputs intermediate predictions. We demonstrate its utility on the SeeAct agent and the Mind2Web benchmark.

3.2 Case Study: SeeAct + Mind2Web

To exemplify our framework, we apply it to the SeeAct web agent (Zheng et al., 2024) and evaluate performance on the multimodal Mind2Web dataset (Deng et al., 2024). Below, we briefly introduce both components and describe the adaptations we made to support modular evaluation.

3.2.1 Mind2Web Dataset

Mind2Web is a benchmark of 2,000 multistep tasks designed for web-based decision making (e.g., "Rent the cheapest SUV starting today"). Each task contains HTML structure, full-page screenshots, and ground-truth user actions. Action include CLICK, TYPE, SELECT, and most tasks require executing multiple sequential steps. One disadvantage of the dataset is, that it accepts (includes) only one single ground-truth path, even though multiple solutions are possible to solve a web-task. To support more flexible and meaningful evaluation, we augment Mind2Web with alternative valid actions (see Section 3.4).

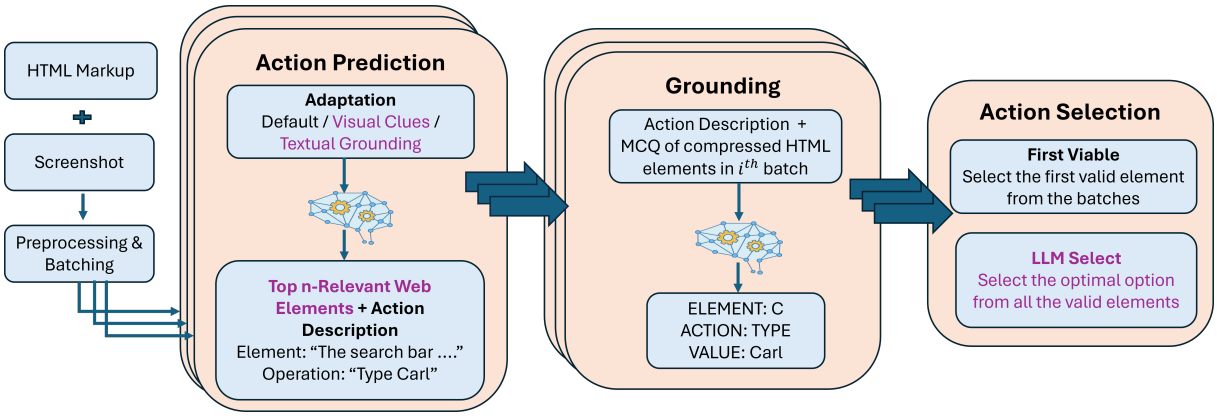

Figure 2: Overview of the adapted SeeAct pipeline. The process consists of three main stages: **Action Prediction** to generate an abstract action description, **Grounding** to map it to a specific HTML element, and **Action Selection** to choose the final command from multiple parallel batch outputs. Our adaptations, such as different **Prompt Templates** (e.g., Textual Grounding), **Intermediate Reasoning** and the final **LLM Select strategy**, are shown within this modular structure.

3.2.2 SeeAct Pipeline

SeeAct is a multimodal agent that uses vision-language models (VLMs) to interpret HTML and screenshots. Given a task instruction and the current web page, it outputs an action triplet *(Element, Action, Value)*. The original pipeline consists of following stages:

- **Preprocessing**: The top-50 task-relevant HTML elements are selected using a fine-tuned ranking model (Deng et al., 2024) and split into several spatial batches for parallel processing.
- **Planning**: A VLM analyzes each batch and predicts the best next action as a natural language description.
- **Grounding**: For each batch, a LLM maps the natural language action description to one of the structured HTML candidates, producing a grounded action triplet. If no suitable match is present, then it selects None.
- **Action Selection**: A simple heuristic selects the first non-"None" grounded action from across all the parallel batches as the final output.

3.3 Adaptations for Fine-Grained Analysis

To better support modular evaluation and address limitations in the original SeeAct pipeline, we introduce several key improvements. We illustrate our full adapted pipeline in Figure 2.

3.3.1 Input Modifications

A key weakness in the original framework is a potential mismatch of candidate elements between the Planning and Grounding stages. To eliminate this, we modified the pipeline to ensure that both stages operate on the identical set of candidate elements. We then created two adaptations to guide the agent's reasoning:

- **Textual Grounding (TG):** The top-k HTML elements are provided to the agent as a textual list with unique letter identifiers, forcing the VLM to operate within a constrained, known candidate set.
- **Visual Clues (VC):** Red bounding boxes highlight the top-k HTML elements directly on the screenshot, constraining the agent's visual focus.

The **Default (Def)** setting serves as our baseline and is defined as the configuration where neither the TG nor VC adaptations are applied.

3.3.2 Enhancing Reasoning and Selection

In addition to the input modifications, we implemented two foundational improvements that are applied across all experiments:

- **Intermediate Reasoning:** We prompt the VLM to first identify and explain up to five relevant elements before it selects a final action. This provides a chain-of-thought-like mechanism for more detailed analysis.
- **LLM-based Action Selection:** We replace the original heuristic ("first viable") selection strategy with a more robust LLM-based selector. Given all grounded candidates from the parallel batches, this module prompts an LLM to compare them and choose the single most plausible action.

3.4 Augmented Evaluation with Flexible Ground Truth

To overcome the limitations of the original Mind2Web labels with only one valid solution, we identify alternative valid actions within a task, focusing on interchangeable steps. For example, when applying two successive filters (for instance to buy black shoes in size 10) "size" or "color" the order does not play any role to finish the task.

We mine such alternatives by checking whether action-relevant HTML elements appear in multiple steps of the same task, using minimal CSS selectors. We treat these as valid substitutes when order does not affect task correctness. Evaluation results are reported with and without these alternatives, providing a more realistic assessment of agent performance.

4 Experimental Setup

We evaluate our approach using a combination of state-of-the-art open-source and proprietary language models, including **Gemini, Claude, InternVL2-LLaMA3** and **GPT-4** variants. All experiments are conducted on a 90-task subset (490 actions) of the multimodal Mind2Web dataset, using three test-set splits - website, domain and task as used by Zheng et al. (2024). Detailed model configurations are provided in Appendix E.

For optimal performance, the VC and Def adaptations are evaluated using four webpage sections (batches), while TG is evaluated using five. Preliminary results for decision-making are provided in Appendix G.

Evaluating natural language outputs from the Def and VC adaptations in the planning stage requires matching free-form text descriptions to specific HTML elements. To perform this robustly, we employ a fine-tuned neural classifier. Specifically, we train a **BGE-Small-en-v1.5** encoder (Xiao et al., 2023) using the SetFit framework (Tunstall et al., 2022) on 800 labeled pairs of HTML-text matches. For the TG adaptation, evaluation is a direct string comparison of the selected identifier (e.g., "C") against the ground truth. Further evaluation details are available in Appendix B.

5 Results & Analysis

We present the results of our modular evaluation, consolidated in Table 2 and Table 3. We first establish the end-to-end performance of the agents to provide a "black box" view. We then leverage our fine-grained framework to diagnose the primary sources of failure within the pipeline, demonstrating the critical insights that a modular approach provides over a single metric.

5.1 Illusion of a single metric: Pinpointing System-Wide Bottlenecks

From a traditional end-to-end perspective ('First Viable' in Table 2), GPT-4o is the top-performing model with 48.78% accuracy. However, this single metric masks the true nature of agent failures. Our modular analysis reveals two primary bottlenecks that systematically limit performance.

The **first major bottleneck is the initial 'Action Prediction' stage (planning)**. Even the best model, GPT-4o, achieves only 70.17% accuracy here, meaning nearly 30% of tasks fail due to flawed initial reasoning before later stages are even attempted. The **second bottleneck is the final 'Action Selection' stage**, where performance further drops. More critically, our analysis of relative performance decline (Figure 3) shows that despite varying absolute scores, all models exhibit a remarkably similar pattern of error propagation across the pipeline. This suggests that the bottlenecks are not model-specific but are systemic challenges inherent to the web navigation task itself, a key insight only visible through modular evaluation.

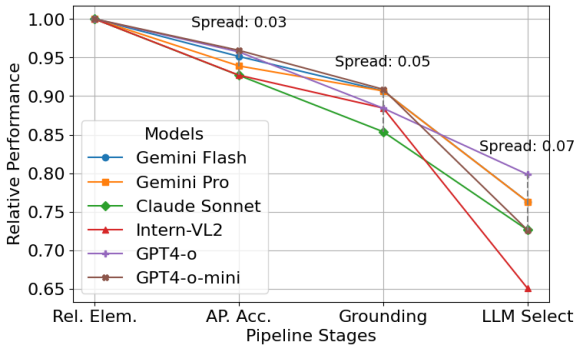

Figure 3: Relative performance decline across pipeline stages for the Def adaptation. Normalizing the initial score reveals a similar drop-off pattern across models, pointing to a shared bottleneck.

5.2 Analysis of Adaptations

To understand the nature of the bottlenecks identified in our pipeline, we now analyze how the TG and VC adaptations influence agent behavior.

The Textual Grounding (TG) adaptation, which reframes element selection as a multiple-

Pipeline Stage:		Action Prediction			Action Selection	
Adaption	Model	RE Acc.	AP Acc.	Grounding	First Viable	LLM Select
Def	Gemini-1.5-flash	61.45	56.59	52.13	28.57	37.76
	Gemini-1.5-pro	62.27	56.18	52.94	35.71	38.57
	Claude 3.5 Sonnet	63.89	56.58	49.27	24.08	36.53
	InternVL2-Llama3-76B-AWQ	61.45	54.16	49.90	17.96	26.53
	GPT-4o	**74.44**	**70.17**	**62.87**	**48.78**	**54.29**
	GPT-4o-mini	61.25	57.18	52.13	28.78	33.88

Table 2: Modular evaluation results for the six models under the standard Default (Def) adaptation, measuring accuracy at each stage of the agent pipeline. This data establishes a performance baseline.

Pipeline Stage:		Action Prediction			Action Selection	
Adaptation	Model	RE Acc.	AP Acc.	Grounding	First Viable	LLM Select
VC	Gemini-1.5-flash	62.89	56.59	47.86	22.24	31.22
	Gemini-1.5-pro	62.48	57.80	48.47	22.65	31.02
	Claude 3.5 Sonnet	63.90	58.62	52.12	21.22	37.76
	InternVL2-Llama3-76B-AWQ	68.96	55.39	51.12	18.98	30.41
	GPT-4o	**78.29**	**75.04**	**67.54**	**44.08**	**53.27**
	GPT-4o-mini	69.57	60.63	50.91	25.51	31.84
TG	Gemini-1.5-flash	80.9	62.01 (67.54)	60.26	21.22	37.35
	Gemini-1.5-pro	79.72	66.74 (70.37)	63.90	24.69	42.24
	Claude 3.5 Sonnet	76.07	63.7 (66.0)	60.45	23.06	37.14
	InternVL2-Llama3-76B-AWQ	82.36	64.31 (72.42)	61.87	21.22	34.49
	GPT-4o	**83.79**	**72.83 (79.7)**	**69.59**	**31.84**	**51.02**
	GPT-4o-mini	73.22	60.47 (67.35)	57.43	18.57	31.22

Table 3: Modular evaluation results for six models across two adaptations Visual Clues (VC) & Textual Grounding (TG), measuring accuracy at each stage of the agent pipeline. As TG employs a separate evaluation pipeline for AP Acc that removes ambiguity, we also present results of the default pipeline (in parentheses) for comparison.

choice question (MCQ), consistently yields the highest Relevant Element and Grounding accuracy. **This confirms that a structured format is highly effective for isolated element identification**. However, this strength reveals a critical trade-off.

The core issue stems from a combination of the binding nature of TG and the agent's lack of global context. Unlike the Def setting where the agent can predict an action on an element not in the candidate set (often leading to a "None" action later), TG forces the agent to choose from the provided list. As each webpage section (batch) is processed in parallel without awareness of others, the agent is prone to selecting a "best-fit" action within nearly every section, even when no action is required. This results in a significantly higher number of viable but unnecessary candidate actions (avg. 3.91 per five sections). **This abundance of options ultimately overwhelms the final Action Selection stage**, explaining why TG's initial gains in identification do not consistently translate to higher final accuracy.

The Visual Clues (VC) adaptation, which provides bounding boxes on the screenshot, yielded highly model-dependent results. While it boosted performance for visually capable models like GPT-4o, it offered negligible benefit—or even harmed performance—for others. This suggests a performance threshold for many models, the cognitive overhead of parsing cluttered visual information and interpreting potentially occluded bounding boxes outweighs the directive benefit of the clues. **This highlights the current limitations in robust, general-purpose visual grounding, where the "help" provided by visual aids can instead become a source of noise**. Details on bounding box occlusion analysis are in Appendix C.

5.3 Analysis of the Final Selection Stage

The final selection stage, where the agent must commit to a single action from a set of grounded candidates, represents the last major hurdle. Across all configurations, **our 'LLM Select' strategy consistently outperforms the simpler 'First Viable' heuristic**, confirming the value of sophisticated reasoning at this final step. However, this stage remains a significant bottleneck.

Our analysis reveals that **performance is di-**

rectly impacted by the number of viable options presented to the selector. Additional information on viable actions and selection performance provided in Appendix D. TG (avg. 3.91) and VC (avg. 2.93) adaptations resulted in an increased number of viable actions compared to the Def (avg. 2.5). As the number of choices increases, selection accuracy across models drops sharply, from an average of 73.1% with two options, down to 56.0% with four. This highlights a critical challenge: the agent's task is not just to identify a correct action, but to disambiguate it from other plausible alternatives.

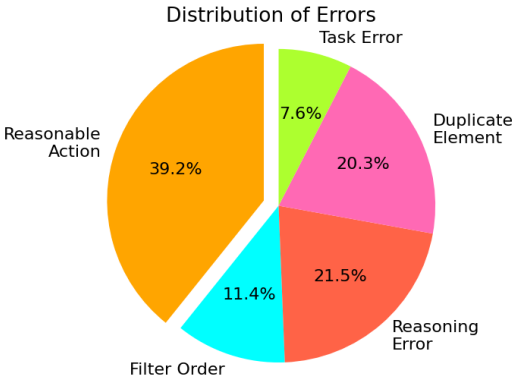

Figure 4: Manual classification of errors for a Gemini-1.5-Pro.

This problem of ambiguity is exacerbated by two factors. First, certain adaptations naturally produce a larger set of viable candidates, increasing the difficulty of the final selection. Second, and more importantly, many of these alternatives are only "incorrect" because of the benchmark's rigid single-ground-truth assumption. Our manual error analysis (Figure 4) confirms this, showing that over 50% of errors are actually Reasonable Actions and Filter Orders.

To quantify the impact of this benchmark rigidity, we augmented the ground truth with a small set of these reasonable mined candidates. As shown in Figure 5, this addition improved performance by up to 8.3%, **directly demonstrating that a significant portion of failure at this stage is attributable to the benchmark's inflexibility**. While our 'LLM Select' strategy handles this ambiguity better than a simple heuristic, these findings underscore the need for more dynamic and flexible evaluation protocols that better reflect real-world web interaction.

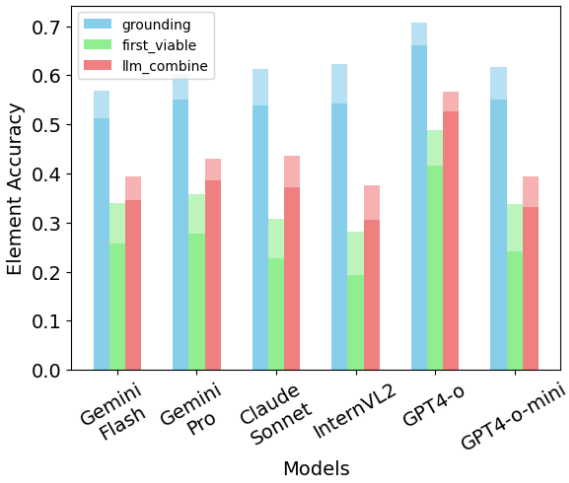

Figure 5: Added performance through additionally mined candidates (Light Shaded)

5.4 Qualitative Error Analysis

To better understand the nature of the failures identified by our modular framework, we manually classified errors from a Gemini-1.5-Pro run on Test-Website split (168 samples) into five categories. The distribution is shown in Figure 4.

- **Reasonable Actions (39.2%):** This category includes semantically correct but non-matching actions, such as clicking on the Nike brand page instead of the "Shoes" category when the task is to buy Nike shoes. Another example is choosing to use the search bar rather than applying filters. While these actions are valid and align with the overall task intent, they deviate from the recorded ground-truth sequence.
- **Alternative Filter Usage and Order (11.4%):** These errors occur when filters are applied in a different order—for example, filtering by color first and then size, or vice versa. We address this issue of filter order variation with our extended ground truth candidate set (Section 3.4).
- **Reasoning Errors (21.5%):** These are genuine failures of logic, where the agent misunderstands the task or the webpage content, such as selecting a clearly incorrect item. This represents the core challenge for improving the models' planning capabilities.
- **Duplicate Element Errors (20.3%):** These errors happen when the agent can't tell apart similar-looking elements, because the compressed HTML doesn't provide enough detail. The agent either selects a semantically similar element or fails due to insufficiently descriptive compressed HTML. This highlights a key weak-

ness in the grounding stage. An example of ambiguous HTML scenarios in Appendix F .

- **Task Errors (7.6%):** These small percentage of errors were attributed to suspected issues in the Mind2Web labeling process where actions don't match the task or issues with reference screenshots.

This qualitative analysis reinforces our main quantitative findings that the two largest sources of failure stem not from trivial errors, but from the agent's difficulty with **benchmark ambiguity** (Reasonable Actions) and **poor grounding/selection signals** (Duplicate Elements).

6 Discussion

Our modular evaluation has uncovered several key insights with broader implications for the design and evaluation of web agents.

Pinpointing the Core Reasoning Failures A key implication of our work is that end-to-end metrics mask the true nature of agent failure. By deconstructing the pipeline, our modular analysis reveals that the agent's journey is bookended by two stages of intense difficulty: initial **Planning** and final **Action Selection**. While errors do occur at every step, these two high-level reasoning tasks represent the primary bottlenecks where the majority of performance is lost. This finding shifts the focus of future research away from uniform pipeline improvements and towards targeted advancements in the agent's core decision-making faculties: forming an effective initial plan and disambiguating the best final action from a set of plausible alternatives.

The Design Tension Our findings on the Textual Grounding (TG) adaptation reveal a critical design tension in parallelized web agents. The architecture's choice to process webpage sections in isolated batches creates a lack of global state, which interacts negatively with structured-input formats like TG. Without inter-batch communication, each batch independently selects its most plausible local action, unable to determine if the true target has already been found elsewhere. This architectural design consistently leads to an over-generation of viable candidate actions. The final 'Action Selection' stage is therefore burdened not with identifying a correct action, but with disambiguating it from a flood of plausible but unnecessary alternatives, which directly results in lower accuracy. This demonstrates that efficiency gains from parallelization can be nullified if the agent lacks a mechanism to maintain a coherent, global understanding of the task.

Rethinking Benchmarking Our analysis, particularly the error classification and the performance boost from mined candidates, compellingly shows that a significant portion of measured "errors" are actually reasonable alternative solutions not captured by the single-ground-truth paradigm of benchmarks like Mind2Web. This has profound implications for the field. It calls for an urgent shift towards more flexible evaluation protocols that can accommodate multiple valid action paths. The continued reliance on rigid benchmarks not only inaccurately penalizes sophisticated models but also steers research away from solving real-world ambiguity. We echo the call for more dynamic, live-web evaluation environments (Zhou et al., 2023) that can provide a more faithful assessment of an agent's true reasoning capabilities.

7 Conclusion

This study demonstrates the utility of modular evaluation for understanding and improving multi-step web agents. Using web navigation as a case study, we adapted the SeeAct framework to trace information flow across stages—from perception to action selection—by introducing targeted input modifications, refined prompting strategies, and a novel LLM-based Action Selection module. Our modular framework supports fine-grained evaluation through both LLM-based and algorithmic metrics.

Experiments across six models on the Mind2Web benchmark not only improved SeeAct's baseline performance but also uncovered key design insights for future web agents:

- **Section-aware reasoning:** Incorporating global page layout and structural context can aid batch-wise perception and decision-making.
- **Visual–semantic grounding:** Tightening the connection between screenshot regions and HTML markup is crucial for robust grounding.
- **Flexible supervision:** Supporting multiple valid actions, rather than relying on a single ground truth, better reflects the ambiguity and flexibility of real-world web tasks.

We hope this evaluation method encourages more interpretable, diagnostic evaluation for complex decision-making agents beyond the web setting.

8 Limitations

Our study demonstrates the value of modular evaluation through a case study on web navigation, but it is limited by its focus on a single framework (SeeAct) and benchmark. While modular evaluation is beneficial for NLP research, it may not apply to agents where intermediate steps lack direct alignment with ground truth. Additionally, although we evaluated six diverse V-LLMs, the inclusion of only one open-source model may underrepresent the performance variety in open-source SOTA models.

References

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2024. Multimodal web navigation with instruction-finetuned foundation models. In *The Twelfth International Conference on Learning Representations*.

Izzeddin Gur, Hiroki Furuta, Austin V 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*.

Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. 2023. Understanding HTML with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2803–2821, Singapore. Association for Computational Linguistics.

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

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024b. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand. Association for Computational Linguistics.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290.

Iat Long Iong, Xiao Liu, Yuxuan Chen, Hanyu Lai, Shuntian Yao, Pengbo Shen, Hao Yu, Yuxiao Dong, and Jie Tang. 2024. OpenWebAgent: An open toolkit to enable web agents on large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 72–81, Bangkok, Thailand. Association for Computational Linguistics.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, 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, page 5295–5306, New York, NY, USA. Association for Computing Machinery.

Eric Li and Jim Waldo. 2024. Websuite: Systematically evaluating why web agents fail. *arXiv preprint arXiv:2406.01623*.

Chang Liu, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Edmund Lam, and Ngai Wong. 2023. Gradually excavating external knowledge for implicit complex question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14405–14417.

Robert Lo, Abishek Sridhar, Frank Xu, Hao Zhu, and Shuyan Zhou. 2023. Hierarchical prompting assists large language model on web navigation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10217–10244, Singapore. Association for Computational Linguistics.

Sahisnu Mazumder and Oriana Riva. 2021. FLIN: A flexible natural language interface for web navigation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2777–2788, Online. Association for Computational Linguistics.

Peter Mühlbacher, Nikos I Bosse, and Lawrence Phillips. 2024. Towards a realistic long-term benchmark for open-web research agents. *arXiv preprint arXiv:2409.14913*.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. 2024. Webcanvas: Benchmarking web agents in online environments. In *Agentic Markets Workshop at ICML 2024*.

Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, and 1 others. 2023. Taskweaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541*.

Mobina Shahbandeh, Parsa Alian, Noor Nashid, and Ali Mesbah. 2024. Naviqate: Functionality-guided web application navigation. *arXiv preprint arXiv:2409.10741*.

Paloma Sodhi, SRK Branavan, Yoav Artzi, and Ryan McDonald. 2024. Step: Stacked llm policies for web actions. *To appear in COLM 2024*.

Hao Sun, Hengyi Cai, Bo Wang, Yingyan Hou, Xiaochi Wei, Shuaiqiang Wang, Yan Zhang, and Dawei Yin. 2023. Towards verifiable text generation with evolving memory and self-reflection. *arXiv preprint arXiv:2312.09075*.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts. *ENLSP Workshop @ NeurIPS 2022*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Cheng Xiong, Gengfeng Zheng, Xiao Ma, Chunlin Li, and Jiangfeng Zeng. 2025. Delphiagent: A trustworthy multi-agent verification framework for automated fact verification. *Information Processing Management*, 62(6):104241.

Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica Lam. 2021. Grounding open-domain instructions to automate web support tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1022–1032, Online. Association for Computational Linguistics.

Tianqi Xu, Linyao Chen, Dai-Jie Wu, Yanjun Chen, Zecheng Zhang, Xiang Yao, Zhiqiang Xie, Yongchao Chen, Shilong Liu, Bochen Qian, Anjie Yang, Zhaoxuan Jin, Jianbo Deng, Philip Torr, Bernard Ghanem, and Guohao Li. 2025. CRAB: Cross-environment agent benchmark for multimodal language model agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21607–21647, Vienna, Austria. Association for Computational Linguistics.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024. Agent lumos: Unified and modular training for open-source language agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12380–12403.

Danqing Zhang, Balaji Rama, Jingyi Ni, Shiying He, Fu Zhao, Kunyu Chen, Arnold Chen, and Junyu Cao. 2025. LiteWebAgent: The open-source suite for VLM-based web-agent applications. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, pages 449–455, Albuquerque, New Mexico. Association for Computational Linguistics.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and 1 others. 2025. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *Agent Learning in Open-Endedness (ALOE) Workshop @ NeurIPS 2023*.

A Full Prompt Example

Table 4 presents the adapted SeeAct prompting scheme used in our work.

B Evaluating Relevant Elements

In this section, we present the pipeline used to evaluate the Action Prediction stage, focusing on the creation of two key metrics: Relevant Element (RE Acc.) and Action Prediction Accuracy (AP Acc). These metrics assess whether the elements involved in or the final action align with the ground truth. Additionally, for the intermediate reasoning stage, we determine whether each listed element is relevant to the task based on the LLM's description.

Due to the non-deterministic structure of the Action Prediction output, we use an LLM (Gemini-1.5-Flash) to extract elements from the reasoning stage. This process generates a JSON object containing the extracted text for each element along with a Boolean value indicating its relevance, as determined by the description in the output. An overview of the pipeline and inputs is shown in Figure 6.

We implemented two variants of the pipeline, depending on the presence of the Textual Grounding

System Role:	Imagine that you are imitating humans doing web navigation for a task step by step. At each stage, you can see the webpage like humans by a screenshot and know the previous actions before the current step decided by yourself through recorded history. You need to decide on the first following action to take. You can click an element with the mouse, select an option, or type text with the keyboard. (For your understanding, they are like the click(), select_option() and type() functions in playwright respectively) One next step means one operation within the three.
Action Prediction:	You are asked to complete the following task: *{task}* Previous Actions: *{prev_actions}* The screenshot below shows a section of a webpage. In this screenshot web elements of interest are outlined with red bounding boxes. Ensure to focus any actions on the highlighted elements. Follow the following guidance to think step by step before outlining the next action step at the current stage: (Current Webpage Identification) Firstly, think about the purpose of this webpage section. Note that you are given section *{batch_id}/{total_batches}* of the webpage. (Previous Action Analysis) Secondly, combined with the screenshot, analyze each step of the previous action history and their intention one by one. Particularly, pay more attention to the last step, which may be more related to what you should do now as the next step. (Web Element Analysis) The screenshot shows *{num_candidates}* web elements (such as links, buttons, and input fields) highlighted with red bounding boxes. Below is a textual description of the *{num_candidates}* highlighted elements: *{choices_simple}* Select up to 5 elements that are most likely to be interacted with based on the current task and previous actions. For each of these 5 elements, describe its general function (e.g., "This date-picker allows the user to select a date for booking a flight.") and explain if interacting with this element is relevant to the task. (Next Action Based on Webpage and Analysis) Then, based on your analysis, in conjunction with human web browsing habits and the logic of web design, decide on the following action. Note that this section of the webpage may contain no viable element to interact with. In this case, you should issue a “None” action. In case there is a viable action clearly outline which element in the webpage users will operate with as the first next target element, its detailed location, and the corresponding operation. To be successful, it is important to follow the following rules: 1. You should only issue a valid action given the current observation. 2. You should only issue one action at a time"""
Action Grounding:	(Reiteration) First, reiterate your next target element, its detailed location, and the corresponding operation. (Multichoice Question) Below is a multi-choice question where the choices correspond to the highlighted elements in the screenshot. The choices are sorted to correspond to their occurrence on the website (top-left to bottom-right). From the screenshot, find out where and what each one is on the webpage. Then, determine whether one matches your target element. Please examine the choices one by one. Choose the matching one. If multiple options match your answer, choose the most likely one by re-examining the screenshot, the choices, and your further reasoning. *{choices}* (Final Answer) Finally, conclude your answer using the format below. Ensure your answer is strictly adhering to the format provided below. Please do not leave any explanation in your answers of the final standardized format part, and this final part should be clear and certain. The element choice, action, and value should be in three separate lines. Format: ELEMENT: The uppercase letter of your choice. ACTION: Choose an action from CLICK, TYPE, SELECT. VALUE: Provide additional input based on ACTION. The VALUE means: If ACTION == TYPE, specify the text to be typed. If ACTION == SELECT, specify the option to be chosen. If ACTION == CLICK, write "None". """

Table 4: Full Prompt Example

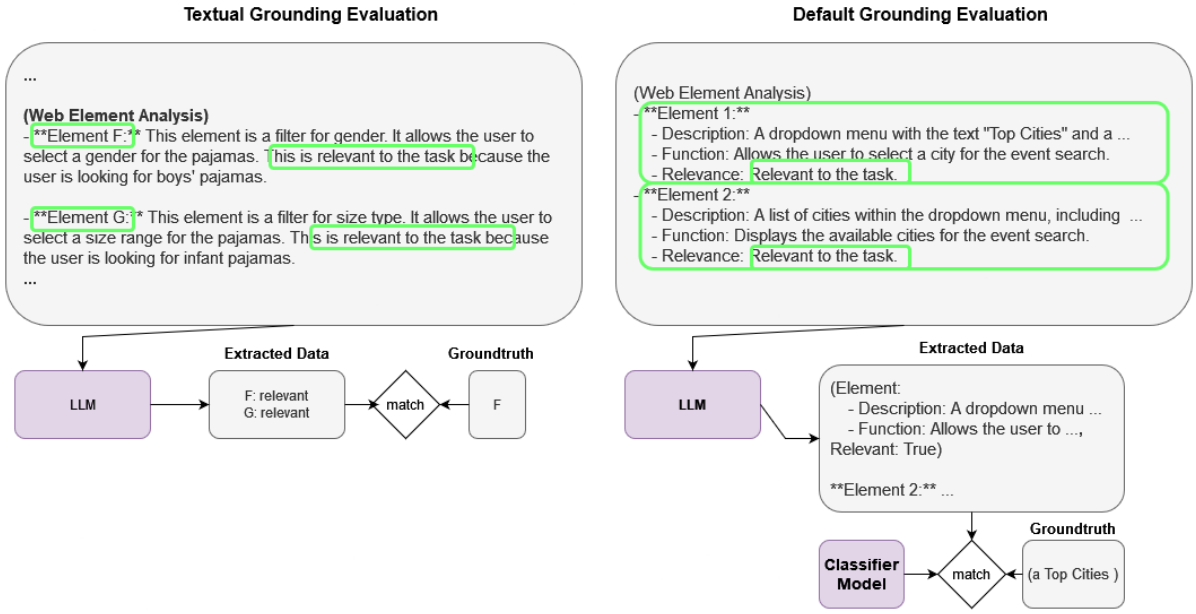

Figure 6: Extraction & Matching pipeline for Action Prediction evaluation

(TG) adaptation. TG simplifies the matching of extracted elements to the ground truth by providing a capital letter identifier, along with a compressed HTML representation, as input during the Action Prediction stage. The LLM detects this identifier (Figure 6), allowing us to match it with the ground truth identifier.

Metric	Value (%)
Accuracy	86.18
Precision	79.28
Recall	82.59
F1 Score	80.09

Table 5: Results of Element matching via classifier

When TG is not enabled, the matching process becomes more complex, as the element descriptions rely solely on visual context, losing any predefined structure. In this case, we still use the LLM to extract listing elements but introduce a secondary classification stage to verify matches. This classifier, BGE-small, was trained using the SetFit framework on 800 manually labeled samples, with 100 additional samples used for evaluation. Each sample consists of an extracted listing element and its corresponding ground truth HTML representation. The 800 samples were drawn from Action Prediction outputs across four different LLMs (Gemini Flash/Pro, GPT 4o/mini) to ensure robustness to structural variations. Evaluation results are presented in Table 5.

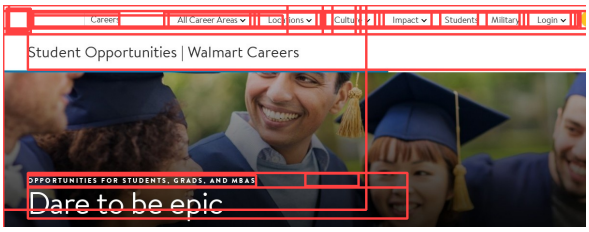

Figure 7: Bounding Box Occlusion Example (parameters chosen to favor occlusion)

We also experimented with using LLMs directly for matching but found their performance to be suboptimal in both zero- and few-shot scenarios.

C Bounding-Box occlusion

The Visual Clues (VC, see Section 3.3.1) adaptation introduces red bounding boxes on webpage images to guide element selection during web navigation tasks.Below we illustrate the problems that can arise from creating bounding boxes based on the coordinates provided by the Mind2Web dataset (Deng et al., 2024). An example screenshot of a webpage section is presented in Figure 7. To provide a clear illustration of these issues, we adjusted pre-processing parameters to favor occlusion; in actual pipeline inference, these effects are typically less pronounced.

Model	TG	VC	Def.
Gemini Flash	4.32	2.99	2.68
Gemini Pro	3.69	2.99	2.34
Claude Sonnet	4.00	2.92	2.71
InternVL2	3.94	3.44	3.24
GPT-4o	3.29	2.34	1.93
GPT-4o-mini	4.23	2.94	2.59

Table 6: Average number of viable action per model and adaptation

	Number of Options		
Model	2	3	4
Gemini Flash	69.43	68.67	58.85
Gemini Pro	72.22	66.5	59.04
Claude	80.15	75.43	60.07
InternVL2	68.9	63.8	51.1
GPT-4o	75.2	68.7	52.8
GPT-4o-mini	72.92	58.39	53.9

Table 7: LLM selection accuracy by number of options

Excessive Nesting of Section Candidates This issue occurs when the ranking model returns multiple hierarchically related elements for the same section. Consequently, multiple tightly packed bounding boxes may overlap, obstructing the content of neighboring web elements. Additionally, the bounding box of a parent container might intersect with surrounding elements, further complicating the visual representation.

Non-Visible Elements Bounding boxes may also be created for elements that do not correspond to visible content in the image. This can potentially confuse the LLM's understanding of the drawn bounding boxes. Such elements include those without visible content (e.g., elements lacking text) or elements that are not currently displayed, such as dropdown menu items within a collapsed dropdown menu.

This section highlights the extra visual understanding required to fully benefit from Visual Clues. While addressing occlusion issues is beyond our scope, preliminary tests suggest that merging neighboring bounding boxes could mitigate them.

D Number of viable Actions

Processing a webpage in multiple sections (batches) causes the Grounding stage to return an equal number of potential actions. These actions can be viable or include a "None" action, resulting from the LLM deciding not to act or a mismatch between the predicted action and the grounding candidate set. In the subsequent Action Selection stage, only viable actions are considered. Therefore, the number of viable actions determines the difficulty of the action selection process by setting the number of options to choose from.

Using the modular evaluation results, we calculated the average number of viable actions across models and adaptations (Table 6).

We find that TG consistently yields the most viable actions, followed by VC and then Def, suggesting that additional constraints during the Action Prediction stage lead to more actions returned. Note that TG was run with 5 sections while VC and Def were run with 4, based on optimal setups for each adaptation. Notably, GPT4-o produces the fewest viable actions across all three adaptations. InternVL2 results in the highest number in two out of three adaptations, closely matching the maximum in the third.

In our second analysis (Table 7), we examined how the number of choices affects action selection accuracy.

We observed that performance declines as the number of actions increases. Claude consistently outperforms other models across all stages, while InternVL2 performs the worst. Since these metrics are based on the modular evaluation results, each model encounters varying numbers of selections with 2, 3, or 4 options depending on prior performance. Excluding Claude as an outlier, selection accuracy with two actions ranges from 68.9% to 75.2%. Indicating that even when faced with only two viable actions models face notable uncertainty.

E Model Specifications

In Table 8 we introduce the specific model versions. Each model was prompted with a temperature of 0.0 to ensure maximal reproducibility during experiments.

For the Open-Source model (InternVL2) a single A100 80GB GPU was utilized for inference.

F Web Element Ambiguity

Below we give a brief overview of three scenarios in which the ambiguous choices influenced the web agents decision making:

Model	Version / Release date
Gemini 1.5 Flash	001 / May 2024
Gemini 1.5 Pro	001 / May 2024
Claude Sonnet	3.5/ 20.06.2024
InternVL2-LLama3 76b	Quantized Version from Huggingface
GPT4o	06.08.2024
GPT4o-mini	18.07.2024

Table 8: Model Versions

- **Identical Compressed HTML**: Multiple buttons with the text "booking" cannot be distinguished by their representation, leading to ambiguity. This calls for including additional context of surrounding HTML elements to uniquely identify each button.

- **Related HTML Elements**: The LLM may choose to interact with a parent element of the ground truth element. While the action would be viable when executed in the browser, it is considered incorrect by Mind2Web's evaluation criteria.

- **Sibling HTML Elements**: A checkbox filter may be accompanied by a link adjacent to it, where clicking either has the same effect. Mind2Web considers only one as the correct action. As SeeACT utilizes a compressed HTML representation (HTML repr.) that highlights salient features of web elements the representation of both may appear similar: (checkbox price range 50) vs. (a price 50)

G Selecting number of webpage sections

Our experiments stretch six models and three adaptations totaling 18 ablations. The main hyperparemter to set in the SeeAct framework (Zheng et al., 2024) is the number of webpage sections (batches) into which the website is split for parallel processing. We decided on using four sections for Default (Def) and Visual Clues (VC) adaptaions as well as five for Textual Grounding (TG). We base this decision on selecting the optimal number of sections through a preliminary result where we ablated 4-5 sections using Gemini-1.5-Flash on the Website split (168 samples). Results of this preliminary study are provided in Table 9. Based on the optimal LLM Select performance we chose the aforementiond number of batches.

Metric	**TG**	**VC**	**Def**
Num Batches = 4			
Grounding	46.15	42.01	46.7
First Viable	22.02	19.6	25
LLM Select	26.7	29.7	36.3
Num Batches = 5			
Grounding	54.51	50	49.41
First Viable	18.45	22.19	22
LLM Select	32.7	28.4	29.76

Table 9: Results for varying number of batches on Website split (168) samples using Gemini-Flash