Modular Agents call for Modular Evaluations

Anonymous ACL submission

Abstract

Current evaluation methods for large language models (LLMs) often overlook the intermediate reasoning steps in multi-step tasks, focusing mainly on final outcomes. This paper proposes a modular evaluation approach for multi-step agent frameworks using web navigation as a case study. We adapt the SeeAct framework by introducing variations in prompts, input modalities, and a new LLM-based action module to better trace information flow. Our approach, tested on an extended version of the Mind2Web dataset, evaluates six different models. Results show that our adaptations outperform the baseline, reveal issues like insufficient global context and ambiguity in HTML processing, and highlight limitations in current offline benchmarks. We provide code for the reimplementation of SeeAct along with our evaluation framework[1].

1 Introduction

Multi-step LLM-based agents are widely employed in complex tasks such as coding (Qiao et al., 2023), text verification (Sun et al., 2023), complex question answering (Liu et al., 2023), and web navigation (Yin et al., 2024; He et al., 2024; Zheng et al., 2024). These frameworks split complex tasks into multiple processing modules, forming a pipeline capable of solving complex problems.

In this work, we focus on web navigation using zero-shot visual large language model (V-LLM) reasoning as a case study. This task is challenging as it uses multiple input types: images, structured markup, and unstructured language. Thus, pipeline components engage different cognitive abilities, like visual and deductive reasoning, enabling us to assess the model's capabilities in these areas by evaluating each one individually. Furthermore, understanding each pipeline component is crucial for identifying weaknesses or systemic issues, yet previous frameworks lack detailed metrics for intermediate reasoning results. Therefore, we advocate for modular evaluations at each intermediate reasoning step.

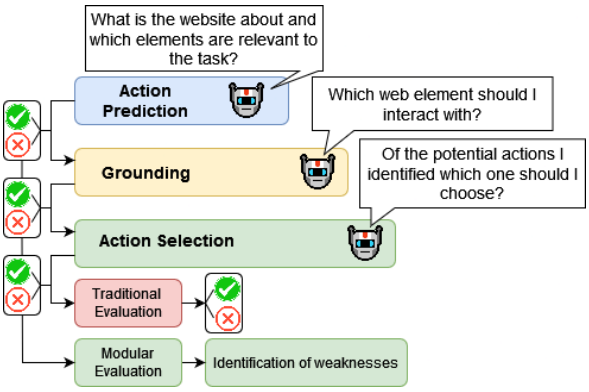

Figure 1: Modular evaluations for the SeeAct Agent.

We select the SeeAct (Zheng et al., 2024) web navigation framework, a four-step pipeline enabling web navigation through zero-shot V-LLM reasoning, as our foundation. To analyze this pipeline, we introduce three adaptations of existing modules and one new module utilizing LLMs for final action selection instead of heuristics. Our adaptations vary the information provided during visual analysis of the webpage by supplying textual descriptions of salient web elements or highlighting them in screenshots. To trace the information flow introduced by these adaptations, we propose a modular evaluation framework that combines LLM and neural classifier-based metrics with algorithmic evaluations. This enables us to verify alignment with ground-truth web elements throughout each pipeline component, including those outputting unstructured natural language reasoning (Figure 1).

We evaluate six closed and open-source models using the Mind2Web dataset (Deng et al., 2024), which contains webpage snapshots with human-labeled actions corresponding to user queries. Each step in the solution process is assigned a single

[1] Anonymous Github Repo

ground-truth action. However, since multiple actions can be valid in real-world scenarios, we address the dataset's limitation by identifying additional viable web elements to provide a more flexible evaluation. Our key contributions are:

- Developed a novel modular evaluation framework for the SeeAct framework, providing insights into intermediate reasoning steps;
- Introduced four adaptations to the SeeAct framework, enhancing the baseline performance and allowing for precise monitoring of information flow within the agent;
- Mined additional ground truth candidates from the Mind2Web dataset and explored the limitations of the offline web navigation benchmark

2 Related Work & Background

2.1 Web Agents

There is a growing focus on developing web agents capable of reasoning and performing action sequences in web applications based on natural language instructions (Gur et al., 2019; Li and Riva, 2021; Mazumder and Riva, 2021; Xu et al., 2021). These agents face the challenge of interpreting complex user inputs and converting them into appropriate web-based actions. However, LLMs can infer next steps from natural language inputs, allowing web agents to perform a broader range of tasks with greater versatility (Furuta et al., 2024; Gur et al., 2023; Sodhi et al., 2024; Wang et al., 2024). Many studies have sought to improve web agents by focusing on HTML document processing (Deng et al., 2024; Gur et al., 2024, 2023; Kim et al., 2024; Lo et al., 2023). Using raw HTML alone often lacks sufficient task-specific context, making action selection difficult.

Recent approaches, such as SeeAct (Zheng et al., 2024), CogAgent (Hong et al., 2024), NaviQAte (Shahbandeh et al., 2024) explore the potential of more advanced LMMs, demonstrating their capacity as generalist web agents. Our case study builds upon these advancements by closely examining how visual and textual information is processed in these agents. Enabling us to identify key areas for improvement.

2.2 Evaluation

Existing work in evaluating web agents primarily focuses on assessing the final outcome of decisions. Some approaches attempt to introduce different assessment metrics for evaluation. For instance, Zhou et al. (2023) introduced functional correctness for task completions, while Li and Waldo (2024) proposed a taxonomy of web actions to identify common failure patterns. Pan et al. (2024) evaluated agents using step and task scores, and Mühlbacher et al. (2024) offered multiple outcome metrics such as *Success* and *Partial Success*. However, these approaches do not break down agent performance at each process step.Shlomov et al. (2024) decomposed tasks into planning and grounding components, evaluating them separately, but only offered element accuracy as a metric.

In contrast, our work introduces a modular evaluation framework that assesses each step of an agent's actions, providing a more granular understanding of performance at different stages of task completion. Thus, it offers insights not provided by existing benchmarks and evaluation methods.

2.3 Mind2Web

For our evaluation, we use the multi-modal Mind2Web web-navigation dataset (Zheng et al., 2024; Deng et al., 2024), which offers real-world websites that are more intricate than the controlled environments in benchmarks like MiniWob++ (Humphreys et al., 2022). The dataset includes 2,000 complex web tasks (e.g. "*Rent the cheapest SUV starting today*"), each annotated with corresponding user actions across various web domains. Annotations are the result of crowd-workers navigating webpages in response to a user query. Further, each action is associated with the corresponding HTML markup, a screenshot of the webpage, a set of ground-truth web elements, and an action type (*CLICK*, *TYPE*, or *SELECT*). Successful task completion is defined by performing a predefined action on a target website.

Mind2Web is traditionally evaluated using four metrics that quantify the selection of the correct web element and action in multiple ways. For our evaluation, we solely focus on Element Accuracy (Elem. Acc.) as this represents the main challenge during web navigation. Element accuracy quantifies whether the selected web element matches the ground truth element.

2.4 SeeAct

SeeAct introduces a multi-modal agent framework relying on the zero-shot reasoning capabilities of V-LLMs to solve web navigation tasks. Each action of the SeeAct framework involves four stages: Data Preprocessing, Action Prediction, Grounding, and

Action Selection. The inputs are the user query, the HTML and the full-page screenshots of the webpage, while the output is a single action triplet.

The data pre-processing stage extracts the top 50 task-relevant HTML elements using a fine-tuned ranking model (Mindact (Deng et al., 2024)), parses them into a compact text format highlighting key features (see compressed HTML Figure 2), and divides them along with the associated screenshot into 4-5 batches based on their webpage location.

Then, in the Action Prediction stage, a V-LLM describes the next best action by analyzing the screenshot associated with each batch. In the Grounding phase, the predicted action description is mapped into a structured action triplet, using an LLM to select the appropriate HTML element from a set of candidates. This includes a "None" option should no fitting candidate be identified. Finally, during Action Selection, one action triplet from the set of webpage sections (batches) is chosen, using a heuristic-based strategy to select the first viable (i.e., not including a "None" element) action.

3 Methodology

This section presents our main contributions, namely pipeline improvements, the introduction of a modular evaluation framework for web-based agents, and an updated evaluation dataset.

3.1 Pipeline Adaptations

In the following, we present the adaptions of the SeeAct web agent pipeline, which concern modifications of existing components and the introduction of a new Action Selection module. Exemplary prompts for the different pipeline stages are in the appendix (Table 6).

Input Modifications In the original SeeAct framework, action prediction relies solely on visual reasoning to identify suitable web elements. This allows the LLM to select actions without constraints. However, the Grounding stage restricts actions to top-k web elements received from a ranking model. Consequently, the unrestricted Action Prediction stage may select a web element not part of the top-k elements used during Grounding, resulting in an invalid action. To address this, we introduce three key modifications to the Action Prediction stage, exemplary also depicted in Figure 2:

- **Textual Grounding (TG)**: The agent is provided the same listing of top-k web elements as used during Grounding. In this listing, each candidate is given a capital letter identifier. This focuses its reasoning and selection efforts on a common set of candidates.
- **Visual Clues (VC)**: To constrain the visual selection space, the top-k web elements are highlighted with red bounding boxes using coordinates from the Mind2Web dataset.
- **Intermediate Reasoning**: Before finalizing an action, the LLM is prompted to identify up to five web elements and explain their relevance to the task.

Intermediate reasoning is applied in all experiments, allowing for a more detailed analysis of the action selection process. Textual Grounding and Visual Clues are ablated to benchmark how LLMs perform under varying levels of information. Should neither VC or TG be utilized, then we call this setting Default (Def).

LLM-based Action Selection In the original SeeAct pipeline, the Action Prediction receives possible actions from the different batches and selects the first viable element as action - which is not necessarily the most suitable. To address this limitation, we introduce an LLM-based reasoning stage (**LLM-select**), in which an LLM receives inputs about viable actions (action description, action triplets and screenshot) and makes a final decision. The output is a single formatted action triplet.

3.2 Modular Evaluation Framework

Our modular evaluation provides insights into how well the agent aligns with the ground truth at each intermediate step. Below, we introduce the metrics for each stage:

Action Prediction In the Action Prediction stage, the agent outputs up to five web elements it deems relevant (see Intermediate Reasoning Section 3.1), along with descriptions of the next best action for each webpage section. By comparing these outputs to the ground truth web element, we can evaluate the agent's ability to assess relevance at different granularities: both within the list of elements and as the selected element for the next action.

Since these outputs are embedded within natural language reasoning, we employ a combination of LLM-based parsing (Gemini 1.5 Flash) and manual parsing to extract the associated web elements. For each element, this process yields a natural language description (e.g., "The search bar with the

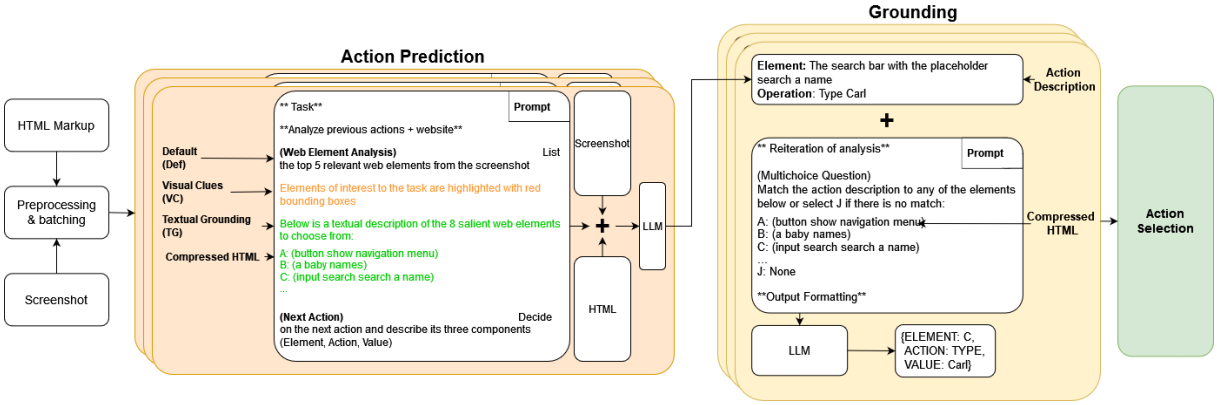

Figure 2: Example of proposed adaptations and prompting in SeeAct

placeholder *'search a name'*.") and, for elements from intermediate reasoning, a relevance rating.

To determine whether these descriptions match the ground truth element, we perform textual matching between the element descriptions and the compressed HTML representation of the ground truth (e.g., (input search a name)). When using the TG adaptation (Section 3.1), we instead match element identifiers provided with the TG inputs since the agent reuses these identifiers when referring to relevant elements.

While matching identifiers simplifies to a boolean check, textual matching requires neural models for validation. We fine-tune a BGE-Small en 1.5 encoder model (Xiao et al., 2023) using the SetFit framework (Tunstall et al., 2022) for efficient low-sample training. Trained on 800 manually labeled pairs of compressed HTML and textual descriptions, this classifier reliably determines whether a ground truth element matches a natural language description. Evaluation results and further evaluation pipeline details are provided in Appendix B.

These steps produce two boolean metrics: `Rel. Elem.` for the intermediate reasoning and `AP. Acc.` for the next best action.

Grounding & Action Selection Since the outputs of Grounding and Action Selection are structured action triplets, matching them to the ground-truth action simplifies to a boolean check. As the Grounding stage involves results of different batches, we only consider the section (batch) where the candidate set includes the ground truth action. Consequently, the `Grounding` metric can also be interpreted as the optimal selection strategy. For Action Selection, this boolean check is applied to the two selection strategies, `First Viable`, and `LLM Select`, as presented above.

3.3 Revised Mind2Web

The Mind2Web dataset assigns a single action sequence to each user query, thus treating alternative sequences, which may achieve the same goal, as errors. To address this, we expand the ground-truth data by identifying additional valid actions, focusing on interchangeable actions. This is common in tasks like filtering items, where the order of applying filters (e.g., "color" or "size") doesn't matter as long as both are applied before the item is selected and both are visible on on the same website.

To implement this, we check if ground-truth elements from one action are present in the HTML of another action of the same task, using minimal CSS selectors. If so, these elements are added as viable alternatives. However, not all sequences allow for interchangeability: for example, a submit button must be clicked after a filter is set. Therefore, we report viable candidates separately to approximate the model's true reasoning capabilities.

4 Experiments & Results

4.1 Experimental Setup

For our experiments we utilize the following mixture of closed source and open source models: Gemini-1.5-Pro, Gemini-1.5-Flash, Claude Sonnet, InternVL2-Llama3, GPT-4o, GPT-4o-mini. Details about specific model versions and parameters are provided in Appendix E.

Modular Analysis Pipeline analysis is conducted using the modular evaluation framework as described in Section 3.2. This experiment solely focuses on Element Accuracy, as it allows us to assess the LLM's capability to grasp the visual and

Pipeline Stage:		Action Prediction			Action Selection	
Model	Adaptation	Rel. Elem.	AP. Acc.	Grounding	First Viable	LLM Select
Gemini-1.5-flash	Def	61.45	56.59	52.13	28.57	37.76
	VC	62.89	56.59	47.86	22.24	31.22
	TG	80.9	62.01 (67.54)	60.26	21.22	37.35
Gemini-1.5-pro	Def	62.27	56.18	52.94	35.71	38.57
	VC	62.48	57.80	48.47	22.65	31.02
	TG	79.72	66.74 (70.37)	63.90	24.69	42.24
Claude 3.5 Sonnet	Def	63.89	56.58	49.27	24.08	36.53
	VC	63.90	58.62	52.12	21.22	37.76
	TG	76.07	63.7 (66.0)	60.45	23.06	37.14
InternVL2-Llama3-76B-AWQ	Def	61.45	54.16	49.90	17.96	26.53
	VC	68.96	55.39	51.12	18.98	30.41
	TG	82.36	64.31 (72.42)	61.87	21.22	34.49
GPT-4o	Def	74.44	70.17	62.87	48.78	54.29
	VC	78.29	75.04	67.54	44.08	53.27
	TG	83.79	72.83 (79.7)	69.59	31.84	51.02
GPT-4o-mini	Def	61.25	57.18	52.13	28.78	33.88
	VC	69.57	60.63	50.91	25.51	31.84
	TG	73.22	60.47 (67.35)	57.43	18.57	31.22

Table 1: Agent Pipeline Analysis for TG (Textual Grounding), VC (Visual Clues), and Def (Default) Adaptations

textual context. We perform pipeline analysis on a subset of the Mind2Web dataset, consisting of 90 tasks (490 actions), taken from Zheng et al. (2024).

For each model, we evaluate three ablations as introduced in Section 3.1: Textual Grounding (TG), Visual Clues (VC), and the default setting (Def). Choosing optimal performance for each we run VC and Def using four webpage sections and TG with five[2]. Each ablation is assessed using five metrics: two for the action-prediction stage (Section 3.2), one for the Grounding stage (Section 3.2), and two for the Action Selection strategies *First Viable* and *LLM-select*, as introduced in Section 3.1. Metrics are provided as the sample weighted average across all three test-set splits (website, domain, task). Except for the Action Selection strategy metrics, all other metrics are the result of the evaluation of the website section (batch), which includes the ground truth element in its candidate set.

4.2 Results

The results of the modular evaluation framework are presented in Table 1.

Adaptations Our analysis reveals that agents are, on average, 8.98% more proficient at identifying the ground truth element (GT Elem.) within relevant elements (Rel. Elem.) than at selecting it as the optimal next action (AP.Acc). This gap is most pronounced in the TG adaptation, reaching 14.33%. Notably, TG achieves the highest Rel. Elem. scores among all adaptations tested.

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. This adjustment improves performance by 5.55%, establishing TG as the best adaptation for AP.Acc across all models (5/6 before).

VC ranks second in both Rel. Elem. and AP.Acc, but its advantage over the default adaptation varies. In lower-performing models (Gemini Flash/Pro, Claude Sonnet), VC improves Rel. Elem. by only 0.55% on average, whereas InternVL2 and GPT models see an average improvement of 6.56%.

For the TG adaptation, TG-based AP.Acc values closely align with Grounding values, differing by just2.76% on average. In contrast, default-based AP.Acc values for TG show a significant drop to Grounding, averaging an 8.31% difference. Models within each adaptation exhibit similar rates of performance decline between AP.Acc and Grounding.

Overall, TG consistently outperforms other adaptations in Grounding performance across all models, while VC surpasses the default adaptation in 3 out of 6 cases.

Action Selection LLM Select outperforms the First Viable selection heuristic across all models and adaptations. Within the LLM Select results the optimal adaptation varies: `Def` leads in 3 out

[2]Preliminary results for decision making in Appendix G

of 6 models, while TG outperforms in 2 out of 6 models. When comparing Grounding (interpreted as optimal selection) to the Action Selection strategies, we observe a significant discrepancy: 29.53% for `First Viable` and 18.56% for `LLM Select`. With respect to models, we find that GPT4-o outperforms its competitors for both `First Viable` and `LLM Select` methods.

4.3 Auxiliary Studies

To provide additional context to our results, we conducted further analyses focusing on action selection performance, viable candidate performance and error propagation.

4.3.1 Action Selection Performance

Action selection performance is influenced by two main factors: the number of available actions and the precision in selecting among them. To provide additional context, we performed an auxiliary analysis of the modular evaluation results[3]. This analysis aimed to determine both the number of viable actions produced during the Grounding stage and the accuracy of selecting the correct action relative to the number of options provided. These metrics can be directly derived from the modular evaluation results without further experimentation.

We found that the TG (avg. 3.91) and VC (avg. 2.93) adaptations resulted in an increased number of viable actions compared to the default (Def) condition (avg. 2.58), with TG producing the highest number.

Regarding the accuracy of selecting the correct action, we calculated accuracy separately based on the number of actions to choose from (ranging from 2 to 4). The results indicate a clear decrease in accuracy as the number of options increases; specifically, the accuracy averaged across models decreases from 73.13% with two options, to 66.91% with three options, and down to 55.96% with four options.

Viable Candidate Performance We present the results of our viable candidate mining (Section 3.3) in Figure 3. These results are averaged across all adaptations. We find that each stage (Grounding, First Viable, and LLM Select) benefits from the additionally mined candidates. The First Viable strategy shows the strongest improvements, with an average increase of 8.3%. However, LLM Select remains the best-performing selection strategy across all models.

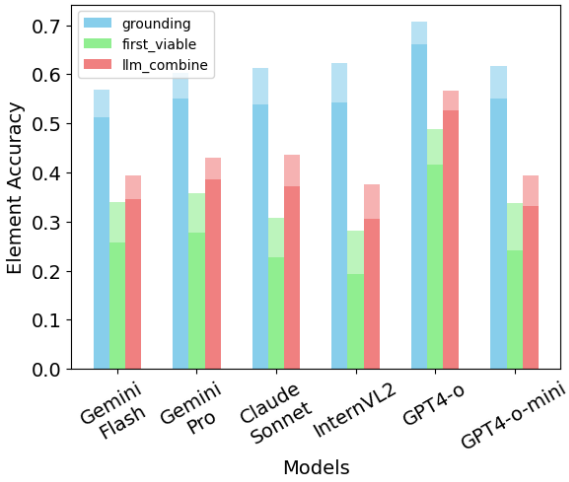

Figure 3: Added performance through additionally mined candidates (Light Shaded)

4.3.2 Error Propagation

Rate of Errors We calculated the performance decline between pipeline stages to assess how errors propagate (e.g. Rel. Elem - AP. Acc = decline). Normalizing each model's Rel. Elem. score to one, we plotted the decline in subsequent stages for the Def adaptation (Figure 4). While absolute performance values differ across models (Table 1), the rates of performance decline between stages are remarkably similar. Excluding InternVL2's LLM-Select performance as an outlier, the spread after accounting for the decline is only 7%, despite a 20.41% variation in actual LLM-Select performance among models.

Types of Errors We manually classified errors from a Gemini-1.5-Pro run on the Test-Website split (168 samples) into five categories (Figure 5).

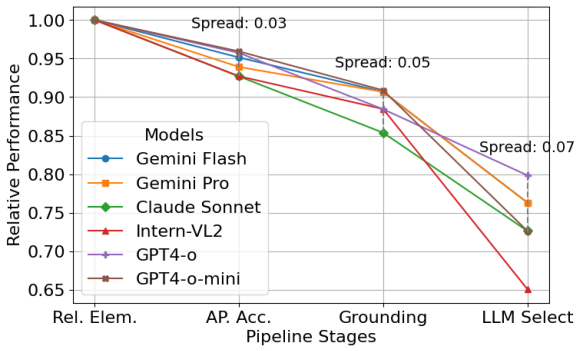

Figure 4: Relative performance across pipeline stages (Def)

[3]Additional information on viable actions and selection performance provided in Appendix D

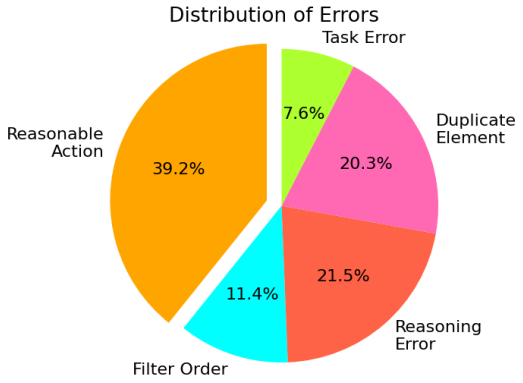

Figure 5: Manual classification of errors for Gemini-1.5-Pro run

Reasonable Action and Filter Order combined account for 50.6% of errors. These involve reasonable actions that differ from the ground truth such as clicking on the Nike brand store instead of the shoe section when the task is to buy Nike shoes, or using a search bar instead of applying filters as well as applying filters in a different order than recorded. The issue of varying filter order is something we address with the extended ground truth candidate set (Section 3.3). The second-largest category, Reasoning Errors, involves clear misunderstandings of the task or the webpage, such as selecting the wrong item. Duplicate Element errors occur when compressed HTML used during grounding lacks detail, leading to incorrect element selection. Lastly, Task Errors stem from assumed errors in the Mind2Web labeling process where actions don't match the task or issues with reference screenshots.

5 Discussion

We organize our discussion into two main parts. First, we contextualize our findings and provide explanations for their observed performance. Second, we discuss the implications of our work for future web agent frameworks and the evaluation of multi-step agents in general.

5.1 Contextualizing Results

Ability to identify non-relevant options In our results, TG consistently achieved the best Grounding performance but ranked best for LLM Select in only 2 out of 6 cases. The reason lies in the number of viable actions returned by the Grounding stage. Here TG produced the most viable actions, thus resulting in lower action selection performance.

Through manual analysis, we found that this increase stems from LLMs' difficulty in choosing not to act, amplified by a lack of global information. In SeeAct, each webpage section is processed in parallel, aware only of its local context. Unlike the Def. setting, which allows the selection of elements outside the candidate set, TG binds Action Prediction to the actual candidates, reducing false positives (i.e. situations in which a "None" action results from a web element chosen during Action Prediction that is outside of the candidate set used during grounding). Without referencing other sections, the agent attempts to perform an action in nearly every section (on average 3.91/5), even when unnecessary.

Ambiguity & additional Options Performance across adaptations is also influenced by two additional factors: viable next actions not recognized by the Mind2Web benchmark and ambiguity in grounding web elements when using compressed HTML representations.

We quantified this ambiguity through two experiments. First, comparing the TG-based and Def-based AP.Acc metrics showed that the Def-based metric (which accepts textually similar elements) increased AP.Acc by 5.55% on average over the TG-based metric that only counts exact matches. This indicates that agents tend to select elements that are semantically similar but not identical to the ground truth elements. Second, manual error labeling revealed that 20.3% of agent errors were due to duplicate elements. Here, the agent either selects a semantically similar element or fails due to insufficiently descriptive compressed HTML[4].

We found ample evidence of LLMs selecting viable next actions that were recorded as errors by the Mind2Web dataset due to its assigning a single ground truth action per task. Specifically, our manual evaluation revealed that 50.6% of errors were reasonable toward completing the task. Our experiments with the extended candidate set, which tackled a specific type of reasonable action (filter order), quantified their effect, showing improvements ranging from 5.48% to 8.3% depending on the pipeline section.

Furthermore, we provided additional evidence of systemic issues in the benchmark by identifying that all models, independent of their reasoning capabilities, face similar rates of decline between pipeline stages despite showing substantial differences in absolute metrics. This does hint at com-

[4]We provide an example of ambiguous HTML scenarios in Appendix F.

mon challenges with samples in the benchmark.

Models & Benefits of Adaptations Our results highlight the benefits of sharing information between related pipeline stages. Each model improved performance up to the Grounding stage when provided with TG-based candidate information, allowing them to focus on relevant candidates. However, offering similar guidance through visual cues (VC) yielded mixed results. While VC improved Rel. Elem. and AP.Acc., it underperformed in the Grounding stage for 3 out of 6 models. We believe this indicates limitations in the models' visual reasoning capabilities. Specifically, InternVL2, GPT-4o, and GPT-4o mini showed a 3.84–8.32% improvement in Rel. Elem., but InternVL2 largely lost this advantage when selecting the next best action (AP. ACC). This suggests that enhanced visual understanding does not compensate for the internal reasoning needed to assess the relevance of web elements. Additionally, bounding boxes cluttered images and possibly occluded important webpage areas[5]. Therefore, for a model to benefit from VC, it must identify salient information despite the added occlusions and visual noise.

5.2 Implications

Zero-Shot Web Agents Our work has four key implications for zero-shot web agents. First, we emphasize the importance of global information to prevent agents from attempting actions in every webpage section. While parallelization speeds up the pipeline, the increase from not sharing information between sections harms output performance.

Second, we highlight the need to guide the initial webpage analysis. In SeeAct, an encoder model ranks the top-k web elements before visual reasoning. Providing this set during Action Prediction boosts the identification of relevant elements. Our results suggest that textual clues (as in TG) are more effective than visual ones, partly due to limitations in visual reasoning and the occlusion caused by bounding boxes.

Third, future frameworks should establish a clear linkage between visual webpage context and HTML markup. We identified ambiguity as a major influence on model performance; even when visual reasoning correctly identifies an element, matching it to the correct HTML markup can be error-prone.

Lastly, while LLMs consistently improve upon heuristic-based action selection, they still fall short of optimal selection accuracy. This deficit may partly stem from benchmark limitations. Improvements could be achieved by aligning with the browsing habits observed in the benchmark, possibly with a trained selector or few-shot prompting.

Web Navigation Benchmarks Our work demonstrates that the single-correct-action approach of the Mind2Web benchmark lacks nuance for assessing true reasoning performance. We recommend switching to benchmarks simulating live web environments (Zhou et al., 2023) when focusing on evaluating true web reasoning performance.

Modular Evaluations Our findings underscore the importance of modular evaluations. Without analyzing each stage of the agent's pipeline, many insights would have remained hidden. In addition to uncovering insights from specific stages, we were able to identify hints of systemic benchmark issues by comparing how the performance of various models deteriorates across all stages.

6 Conclusion

In this study, we underscored the value of modular evaluations for multi-step agents, using web navigation as a case study. Building upon the existing web navigation framework SeeAct (Zheng et al., 2024), we introduced a series of adaptations aimed at investigating the information flow from initial webpage analysis to final action selection. These adaptations included changes to input modalities and prompting, along with the development of a novel LLM-based Action Selection stage. Our modular framework enabled us to evaluate each individual step taken by the agent, leveraging a combination of LLM-based and algorithmic metrics. Evaluating six closed- and open-source models on the Mind2Web web navigation dataset (Deng et al., 2024), we not only improved upon the SeeAct baseline but also uncovered critical insights for the advancement of web agents. Specifically, we identified three key points to improve by: a) Incorporating global information when processing webpage sections so each part is informed by the others; b) Strengthening the linkage between visual webpage context and HTML markup for reliable mapping of reasoning to actions.; c) Loosening the single-action ground truth restriction of Mind2Web and allowing for multiple potential actions.

[5]Additional information on bounding box occlusion provided in Appendix C

7 Limitations

Our research established the value of modular evaluation based on a case study of web navigation. Our study is naturally limited by the scope of our evaluation, which is based on a single web navigation framework (SeeAct) and tested on a single web navigation benchmark. While we are certain about the value of enhanced evaluation measures for the state of NLP research, we acknowledge that a modular evaluation framework like ours may not be applicable for agents where intermediate steps share no direct information about their alignment with the ground truth.

While we evaluate a diverse range of six V-LLM, we acknowledge that we have only considered a single open-source model. Thus, the variety of performance of the open-source SOTA may be underrepresented.

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.

Izzeddin Gur, Ulrich Rueckert, Aleksandra Faust, and Dilek Hakkani-Tur. 2019. Learning to navigate the web. In *International Conference on Learning Representations*.

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

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 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.

Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. 2022. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pages 9466–9482. PMLR.

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

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

Yuanchun Li and Oriana Riva. 2021. Glider: A reinforcement learning approach to extract ui scripts from websites. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1420–1430, New York, NY, USA. Association for Computing Machinery.

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, et al. 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*.

Segev Shlomov, Aviad Sela, Ido Levy, Liane Galanti, Roy Abitbol, et al. 2024. From grounding to planning: Benchmarking bottlenecks in web agents. *arXiv preprint arXiv:2409.01927*.

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, et al. 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.

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.

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.

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, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *Agent Learning in Open-Endedness (ALOE) Workshop @ NeurIPS 2023*.

A Full Prompt Example

Table 2 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 (Rel Elem) and Action Prediction Accuracy (AP. Acc). These metrics assess whether the elements involved in intermediate reasoning 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 (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.

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

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 2: Full Prompt Example

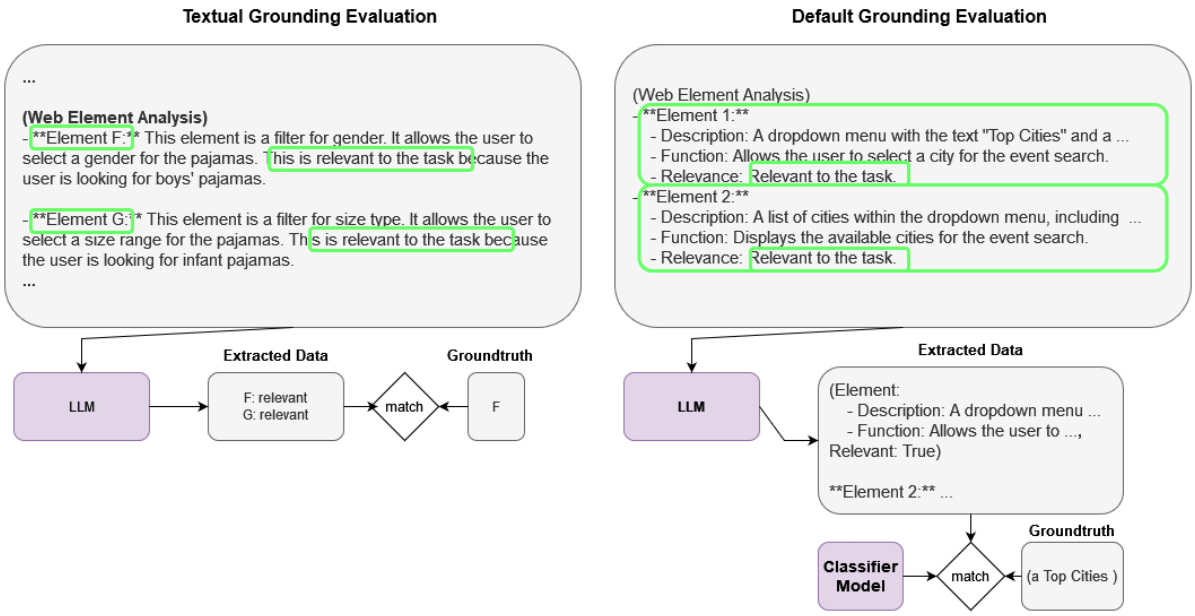

Figure 6: Extraction & Matching pipeline for Action Prediction evaluation

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

Table 3: Results of Element matching via classifier

structural variations. Evaluation results are presented in Table 3.

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 3.1) 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.

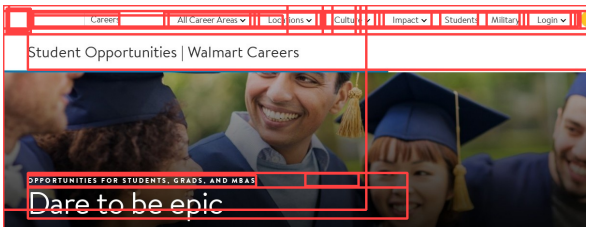

Figure 7: Bounding Box Occlusion Example (parameters chosen to favor occlusion)

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 under-

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 4: 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 5: LLM selection accuracy by number of options

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 6: Model Versions

standing 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 4).

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 **??**), 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 6 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:

- **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 7. 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 7: Results for varying number of batches on Website split (168) samples using Gemini-Flash