Beyond Syntax: Action Semantics Learning for App Agents

Supplementary Material

6. Statement on LLM Usage

We disclose that large language model (LLM) tools were used solely for language refinement, including improving grammar and polishing phrasing. LLMs were not used to generate scientific content, research ideas, experiment designs, data, analyses, or code. All suggestions and modifications from these tools were made under the direct supervision and final approval of the authors, and all authors are fully aware of and consent to this usage.

7. Experimental Environment

For the experiments of Online Dataset Comparison and Offline Dataset Comparison, all training is conducted on NVIDIA V100 GPUs. Our online tests follow the instructions of the original benchmark datasets. Specifically, we evaluate our App agent on the Android Virtual Device (AVD) with system images Android 13.0 Tiramisu (API Level 33) following the setup of prior work [16, 30, 40].

For the experiment of RL Fine-tuning with SEE, each run on the AitW tasks with Filtered BC-SEE and DigiRL-SEE takes approximately 24 hours on a single NVIDIA GeForce RTX 4090 GPU and 8 Intel Xeon CPUs. The online interaction and evaluation is conducted on the AVD with system images Android 9.0 Pie (API Level 28), following the setup of prior work [2, 39]. On the WebArena-Lite benchmark, each run of Filtered BC-SEE, DigiRL-SEE, and WebRL-SEE requires about 24 hours on 8 NVIDIA GeForce RTX 4090 GPUs and 8 Intel Xeon CPUs.

8. Proof for Theorem 3.1

Proof. For the App agent trained by minimising the syntax learning objective, we have:

$$P(\text{success}) = \prod_{i=1}^{n} p_\theta\big(a_i^* \mid g_i, h_i, o_i\big) = \gamma^n,$$

where $\gamma \to 1$. Let $\tilde{o}_i = \delta(o_i)$ be the perturbed observation that never appears in the training set, which makes the correct action satisfying $\delta(a_i^*) \neq a_i^*$. Since the optimiser has no incentive to allocate *more* probability to any specific action under $\tilde{o}_i$ than under o_i, we can have:

$$p_\theta\big(\delta(a_i^*) \mid g_i, h_i, \tilde{o}_i\big) \leq p_\theta\big(\delta(a_i^*) \mid g_i, h_i, o_i\big).$$

Theron, we have:

$$\begin{aligned} P(\text{success} \mid \delta) &= \prod_{i=1}^{n} p_\theta\big(\delta(a_i^*) \mid g_i, h_i, \tilde{o}_i\big) \\ &\leq \prod_{i=1}^{n} p_\theta\big(\delta(a_i^*) \mid g_i, h_i, o_i\big) = (1-\gamma-\eta)^n. \end{aligned}$$

where $\eta \in (0, 1-\gamma)$. Since $\gamma \to 1$, we have $\gamma > (1-\gamma-\eta)$. $P(\text{success}) > P(\text{success} \mid \delta)$ and $\Delta(\delta) = P(\text{success}) - P(\text{success} \mid \delta) > 0$ as claimed. □

9. Proof for Theorem 3.2

Proof. Let $\mathcal{A}^{eq}_{a_i^\star}$ be the semantics equivalent space for the ground-truth action $a_i^\star$. Moreover, since the given perturbation is a semantically preserving but syntactically non-trivial perturbation, we have: $\forall i, |\mathcal{A}^{eq}_{a_i^\star}| \geq 2$. Assume that the semantics-aware agent π_θ^{S} achieves an expected reward within ε of the optimal, i.e., $J(\theta) \geq J^* - \varepsilon$, where J^* is the maximum achievable reward. By Markov's inequality, the probability that the agent selects an action with reward less than $1-\sqrt{\varepsilon}$ is at most $\sqrt{\varepsilon}$. Therefore, for each step i, the probability of selecting a semantically equivalent action under perturbation δ is at least $1-\sqrt{\varepsilon}$, where $\sqrt{\varepsilon} \to \frac{1}{|\mathcal{A}^{eq}_{a_i^\star}|} \leq \frac{1}{2}$. Assuming independence across steps, we can have:

$$P^{\mathrm{S}}(\text{success} \mid \delta) \geq (1-\sqrt{\varepsilon})^n.$$

On the other hand, the syntax-only agent $\pi_\theta^{\mathrm{SFT}}$, trained to minimise cross-entropy loss, assigns $\gamma \to 1$ to the exact training action a_i^* and low probability to other actions. Under perturbation δ, where $\delta(a_i^*) \neq a_i^*$, the agent's probability of selecting the correct action decreases to at most $1-\gamma$ per step. Assuming independence across steps, we can have:

$$P^{\mathrm{SFT}}(\text{success} \mid \delta) \leq (1-\gamma)^n.$$

Given that $0 < \sqrt{\varepsilon} < \gamma < 1$, it follows that $(1-\sqrt{\varepsilon})^n > (1-\gamma)^n$, implying that $P^{\mathrm{S}}(\text{success} \mid \delta) > P^{\mathrm{SFT}}(\text{success} \mid \delta)$. Therefore, $P^{\mathrm{S}}(\text{success} \mid \delta) - P^{\mathrm{SFT}}(\text{success} \mid \delta) > 0$. □

10. Implementation Details

10.1. Dataset

AndroidWorld includes a high-fidelity simulator that emulates realistic Android device behaviour, enabling interactions with 20 real-world applications across 116 user-defined tasks. To determine whether a task is successfully completed, each task is paired with manually authored rules that

provide precise success criteria. *AndroidLab* includes 138 predefined tasks across nine Android applications built on virtual devices. Each task is divided into multiple required page states as sub-goals, with UI tree structure matching verifying correct traversal. The AndroidLab environment ensures reproducibility and eliminates external network or time dependencies. *AndroidControl* comprises 15,283 human demonstrations spanning 14,548 unique tasks across 833 Android applications. *Android in the Wild (AitW)* is a large-scale dataset containing 715k human demonstrations across 30k unique natural language instructions and 357 apps, capturing realistic multi-step interactions on Android devices and websites. *WebArena-Lite* is a human-verified subset from WebArena, which contains 165 tasks spanning diverse websites such as Gitlab, maps, forums, shopping, and CMS, designed to evaluate multimodal web agents under realistic and complex user instructions .

10.2. RL Fine-tuning

Supervised fine-tuning uses human-annotated ground-truth actions to compute the semantic score between the predicted and ground-truth actions. In contrast, for RL fine-tuning, no human annotations are available. Instead, we prompt Gemini [6] as a teacher model to predict an action and adopt it as the reference action, since Gemini demonstrates high semantic fidelity and robustness, making it a suitable proxy for human-annotated actions.

In the experiments on AitW tasks, we evaluate the effect of our SEE module against various methods, including AppAgent [46], CogAgent [12], AutoUI [48], Filtered BC [25], and DigiRL [2]. For Filtered BC, Filtered BC-SEE, DigiRL, and DigiRL-SEE, all experiments are initialised from the AutoUI-Base model, ensuring a consistent starting point across methods. In the experiments on the WebArena-Lite benchmark, the compared methods include SFT, Filtered BC, DigiRL, and our SEE-augmented counterparts. All models are initialised from the SFTed Llama-3.1-8B model released by WebRL [28], which was trained on the trajectory data collected from the WebArena-Lite benchmark.

For the implementation of Filtered BC-SEE, we filters out the successful trajectories $\mathcal{D}_{\text{succ}}$ and applies behavior cloning to the ground-truth action $a_i^\star$. We then extend the filtered BC loss with a semantics-aware term:

$$\ell_\theta^S(g_i, h_i, o_i, \tilde{a}_i) = -\,\tilde{r}(a_i, \tilde{a}_i, g_i, h_i, o_i)\,\log p_\theta(a_i \mid g_i, h_i, o_i), \tag{12}$$

and the loss function of Filtered BC-SEE is:

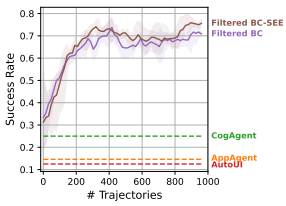
$$\ell_\theta^{\text{FBC-SEE}}(g_i, h_i, o_i, a_i^\star, \tilde{a}_i) = \ell_\theta^S(g_i, h_i, o_i, \tilde{a}_i) + \lambda_i\,\ell_\theta^{SFT}(g_i, h_i, o_i, a_i^\star) \tag{13}$$

where the original filtered BC loss $\ell_\theta^{SFT}(g_i, h_i, o_i, a_i^\star) = -\log p_\theta(a_i^\star \mid g_i, h_i, o_i)$, $\lambda_i = 1 - (1-\alpha_i)\,\tilde{r}$, and $\alpha_i = \exp(-\ell_\theta^{SFT}(g_i, h_i, o_i, a_i^\star))$ is defined in Eq. 5.

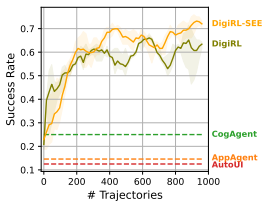
The VLMs used in the experiments of RL fine-tuning are summarised in Table 8, and the hyperparameter settings are provided in Table 9 and Table 10 respectively.

11. Training Curves

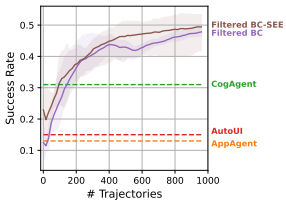
To further illustrate the effect of our SEE module, we present the training curves on the AitW General and Web Shopping tasks under both the Filtered BC and DigiRL optimisation settings in Fig. 3 and Fig. 4 respectively. The curves compare the baseline methods with their SEE-augmented counterparts, showing how semantic-level feedback influences convergence dynamics.



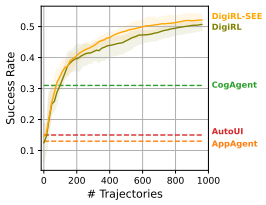
(a) General: Filtered BC vs. Filtered BC-SEE



(b) General: DigiRL vs. DigiRL-SEE



(c) Web Shopping: Filtered BC vs. Filtered BC-SEE



(d) Web Shopping: DigiRL vs. DigiRL-SEE

Figure 3. Training curves on AitW General and Web Shopping tasks. In all cases, incorporating our semantic estimator (SEE) leads to faster convergence and higher final success rates compared with the corresponding baselines.

12. Additional Experiment

In this section, we present experiments on the AndroidControl random-500 setup [16], comparing our method with models including Aria-UI [43], GUI-R1 [22], and AGUVIS [41]. As shown in Table 11, our approach outperforms these models.

13. Prompt

13.1. Prompt for T3A*

```
You are an agent who can operate an Android phone
on behalf of a user.  Based on user's
goal/request, you may
- Answer back if the request/goal is a question
(or a chat message), like user asks "What is my
schedule for today?"
```

Table 8. Summary VLMs used in RL fine-tuning.

Component	Task		Description
	AitW	WebArena-Lite	
Actor Model	AutoUI-Base	Llama-3.1-8B	Serves as the current policy for generating rollout actions.
Teacher Model	Gemini-2.5-Flash	Gemini-2.5-Flash	Provides proxy actions to replace costly human annotations.
World Model	Gemini-2.5-Flash	Gemini-2.5-Flash	Predicts the state transition given an action.
Evaluator	Gemini-1.5-Pro	Llama-3.1-8B	Automatically evaluates the success of tasks.

Table 9. Hyperparameter settings for the experiments on the AitW tasks.

Hyperparameter	Value
Batch Size	4
Total Trajectories	1000
Learning Rate	1e-4
Update Epoch (Actor)	20
Update Epoch (Critic)	5
Maximum Gradient Norm	0.01
Semantic Threshold (τ)	0.6
Discount Factor (η)	0.5
Semantic Reward Weight (β)	0.5

Table 10. Hyperparameter settings for the experiments on the WebArena-Lite benchmark.

Hyperparameter	Value
Batch Size	128
Total Trajectories	1000
Learning Rate	1e-6
Update Epoch (Actor)	1
Update Epoch (Critic)	1
Maximum Gradient Norm	1.0
Semantic Threshold (τ)	0.6
Discount Factor (η)	0.9
Semantic Reward Weight (β)	0.5

Table 11. Comparison on AndroidControl random 500 setup [16].

Method	Step.Acc
Aria-UI	10.2
GUI-R1-3B	46.6
GUI-R1-7B	51.7
AGUVIS-7B	61.5
AGUVIS-72B	66.4
ASL-7B (Ours)	**68.9**

```
- Complete some tasks described in the
requests/goals by performing actions (step by
step) on the phone.
When given a user request, you will try to
complete it step by step. At each step, a list
of descriptions for most UI elements on the
current screen will be given to you (each element
```

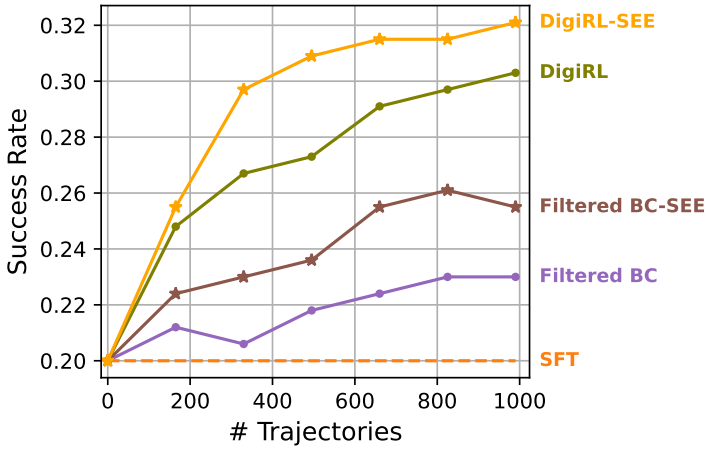


Figure 4. Training curves on the WebArena-Lite benchmark. Compared to the original baselines, incorporating our SEE helps achieve smoother optimisation dynamics and higher final task success rates, further confirming the effectiveness of semantic-level feedback in web control.

```
can be specified by an index), together with a
history of what you have done in previous steps.
Based on these pieces of information and the goal,
you must choose to perform one of the action in
the following list (action description followed
by the JSON format) by outputing the action in
the correct JSON format.
- If you think the task has been completed, finish
the task by using the status action with complete
as goal_status: {{"action_type": "status",
"goal_status": "complete"}}
- Answer user's question:" {{"action_type":
"answer", "text": "<answer_text>"}}
- Click/tap on a UI element (specified by its
index) on the screen: {{"action_type": "click",
"index": <target_index>}}.
- Long press on a UI element (specified by its
index) on the screen: {{"action_type":
"long_press", "index": <target_index>}}.
- Clear the text in a text field (specified by its
index): {{"action_type": "clear_text", "index":
<target_index>}}
- Type text into an editable text field (specified
by its index), this action contains clicking the
text field, typing in the text and pressing the
enter, so no need to click on the target field to
start: {{"action_type": "input_text", "text":
<text_input>, "index": <target_index>}}
- Press the Enter key: {{"action_type":
"keyboard_enter"}}
- Navigate to the home screen: {{"action_type":
"navigate_home"}}
- Navigate back: {{"action_type":
"navigate_back"}}
```

- Scroll the screen or a scrollable UI element in one of the four directions, use the same numeric index as above if you want to scroll a specific UI element, leave it empty when scroll the whole screen: {{"action_type": "scroll", "direction": <up, down, left, right>, "index": <optional_target_index>}} Notably, scroll left to reveal more on the left, scroll right to reveal more on the right, scroll up to reveal more above, and scroll down to reveal more below.
- Open an app (nothing will happen if the app is not installed): {{"action_type": "open_app", "app_name": <name>}}
- Wait for the screen to update: {{"action_type": "wait"}}

The overall user goal/request is: {goal}
Here is a history of what you have done so far: {history}
Here is a list of descriptions for some UI elements on the current screen: {ui_elements_description}
Here are some useful guidelines you need to follow:
General
- Usually there will be multiple ways to complete a task, pick the easiest one. Also when something does not work as expected (due to various reasons), sometimes a simple retry can solve the problem, but if it doesn't (you can see that from the history), try to switch to other solutions.
- Sometimes you may need to navigate the phone to gather information needed to complete the task, for example if user asks "what is my schedule tomorrow", then you may want to open the calendar app (using the 'open_app' action), look up information there, answer user's question (using the 'answer' action) and finish (using the 'status' action with complete as goal_status).
- For requests that are questions (or chat messages), remember to use the 'answer' action to reply to user explicitly before finish! Merely displaying the answer on the screen is NOT sufficient (unless the goal is something like "show me ...").
- If the desired state is already achieved (e.g., enabling Wi-Fi when it's already on), you can just complete the task.
Action Related
- Use the 'open_app' action whenever you want to open an app (nothing will happen if the app is not installed), do not use the app drawer to open an app unless all other ways have failed.
- Use the 'input_text' action whenever you want to type something (including password) instead of clicking characters on the keyboard one by one. Sometimes there is some default text in the text field you want to type in, remember to delete them before typing.
- For 'click', 'long_press' and 'input_text', the index parameter you pick must be VISIBLE in the screenshot and also in the UI element list given to you (some elements in the list may NOT be visible on the screen so you can not interact with them).
- Consider exploring the screen by using the 'scroll' action with different directions to reveal additional content.

- The direction parameter for the 'scroll' action can be confusing sometimes as it's opposite to swipe, for example, to view content at the bottom, the 'scroll' direction should be set to "down". It has been observed that you have difficulties in choosing the correct direction, so if one does not work, try the opposite as well.
Text Related Operations
- Normally to select some text on the screen: <i> Enter text selection mode by long pressing the area where the text is, then some of the words near the long press point will be selected (highlighted with two pointers indicating the range) and usually a text selection bar will also appear with options like 'copy', 'paste', 'select all', etc. <ii> Select the exact text you need. Usually the text selected from the previous step is NOT the one you want, you need to adjust the range by dragging the two pointers. If you want to select all text in' the text field, simply click the 'select all' button in the bar.
- At this point, you don't have the ability to drag something around the screen, so in general you can not select arbitrary text.
- To delete some text: the most efficient way is to directly use the 'clear_text' action, which removes all content from a specific text field in one step. Alternatively, the traditional method is to place the cursor at the right position and use the backspace button on the keyboard to delete characters one by one (long pressing backspace can accelerate this if there are many characters). Another approach is to first select the text you want to delete, then press the backspace button on the keyboard.
- To copy some text: first select the exact text you want to copy, which usually also brings up the text selection bar, then click the 'copy' button in bar.
- To paste text into a text box, first long press the text box, then usually the text selection bar will appear with a 'paste' button in it.
- When typing into a text field, sometimes an auto-complete dropdown list will appear. This usually indicating this is a enum field and you should try to select the best match by clicking the corresponding one in the list.
Now output an action from the above list in the correct JSON format, following the description of the action you are taking. Your answer should look like:
Description: ...
Action: {{"action_type":...}}
Your Answer:

13.2. Prompt for World Model

You are an intelligent autonomous agent designed to predict how a smartphone UI will change in response to specific user actions. You will be given the following:
1. a list of descriptions for most UI elements on the current screen. 2. action1: a user action that to be applied to the current UI. 3. action2: a user action that to be applied to the current UI. action 1 and action 2 are individual actions. Each action is composed of an action type and a corresponding parameter. If the parameter is an index, it refers to the index of a UI element in the provided list.
Additionally, assess whether the two actions would result in the same or similar UI changes. Provide a similarity score between 0 and 1 (with two decimal places), where 0 means completely different and 1 means identical.
Your Task: For each user action, analyze the current UI and accurately predict the resulting change. Describe each predicted change clearly and in detail.

```
Here is a list of descriptions for some UI
elements on the current screen:  ui The user
actions are:  "action1":pred,"action2":label'
**Output Format**:  Return only predicted UI
changes as a JSON map of strings as following.
Do not include any additional explanations,
formatting, or comments.
"action1":"change1...","action2":"change2...",
"score":"similarity_score"'
```

13.3. Prompt for Teacher Model in the experiments on AitW

```
Given a mobile screen and the current context,
provide the next action based on the screen
information.
Available Actions (choose one):  1.  Dual-point
gesture (click):  {{"action_type":  "DUAL_POINT",
"touch_point":  "[y_coord, x_coord]",
"lift_point":  "[y_coord, x_coord]", "typed_text":
""}} - For clicking:  touch_point and lift_point
should be the same coordinates - Coordinates are
normalized between 0.0 and 1.0 (e.g., [0.5, 0.3]
means 50% down, 30% right) - Use 4 decimal places
for precision (e.g., [0.7761, 0.7089])
2.  Dual-point gesture (scroll):  {{"action_type":
"DUAL_POINT", "touch_point":  "[start_y,
start_x]", "lift_point":  "[end_y, end_x]",
"typed_text":  ""}} - For scrolling up:
{{"touch_point":  "[0.8, 0.5]", "lift_point":
"[0.2, 0.5]"}} - For scrolling down:
{{"touch_point":  "[0.2, 0.5]", "lift_point":
"[0.8, 0.5]"}} - For scrolling left:
{{"touch_point":  "[0.5, 0.8]", "lift_point":
"[0.5, 0.2]"}} - For scrolling right:
{{"touch_point":  "[0.5, 0.2]", "lift_point":
"[0.5, 0.8]"}}
3.  Type text:  {{"action_type":  "TYPE",
"touch_point":  "[-1.0, -1.0]", "lift_point":
"[-1.0, -1.0]", "typed_text":  "<your_text>"}}
4.  Navigate back:  {{"action_type":
"PRESS_BACK", "touch_point":  "[-1.0, -1.0]",
"lift_point":  "[-1.0, -1.0]", "typed_text":
""}}
5.  Navigate home:  {{"action_type":
"PRESS_HOME", "touch_point":  "[-1.0, -1.0]",
"lift_point":  "[-1.0, -1.0]", "typed_text":
""}}
6.  Press enter:  {{"action_type":  "PRESS_ENTER",
"touch_point":  "[-1.0, -1.0]", "lift_point":
"[-1.0, -1.0]", "typed_text":  ""}}
7.  Task complete:  {{"action_type":
"STATUS_TASK_COMPLETE", "touch_point":  "[-1.0,
-1.0]", "lift_point":  "[-1.0, -1.0]",
"typed_text":  ""}}
Current Context:  {processed_obs}
Screen:  <See the attached mobile screen image>
Instructions:  - Analyze the mobile screen
carefully - Choose the most appropriate action to
progress towards the goal - For clicks, estimate
the center coordinates of the target element -
For scrolls, choose appropriate direction and
distance - For typing, provide the exact text to
be entered - Output only the JSON action format,
no additional text
Answer:
```

14. Qualitative Examples

We provide qualitative examples of our SEE modules applied to AitW tasks in Fig. 5 and Fig. 6.

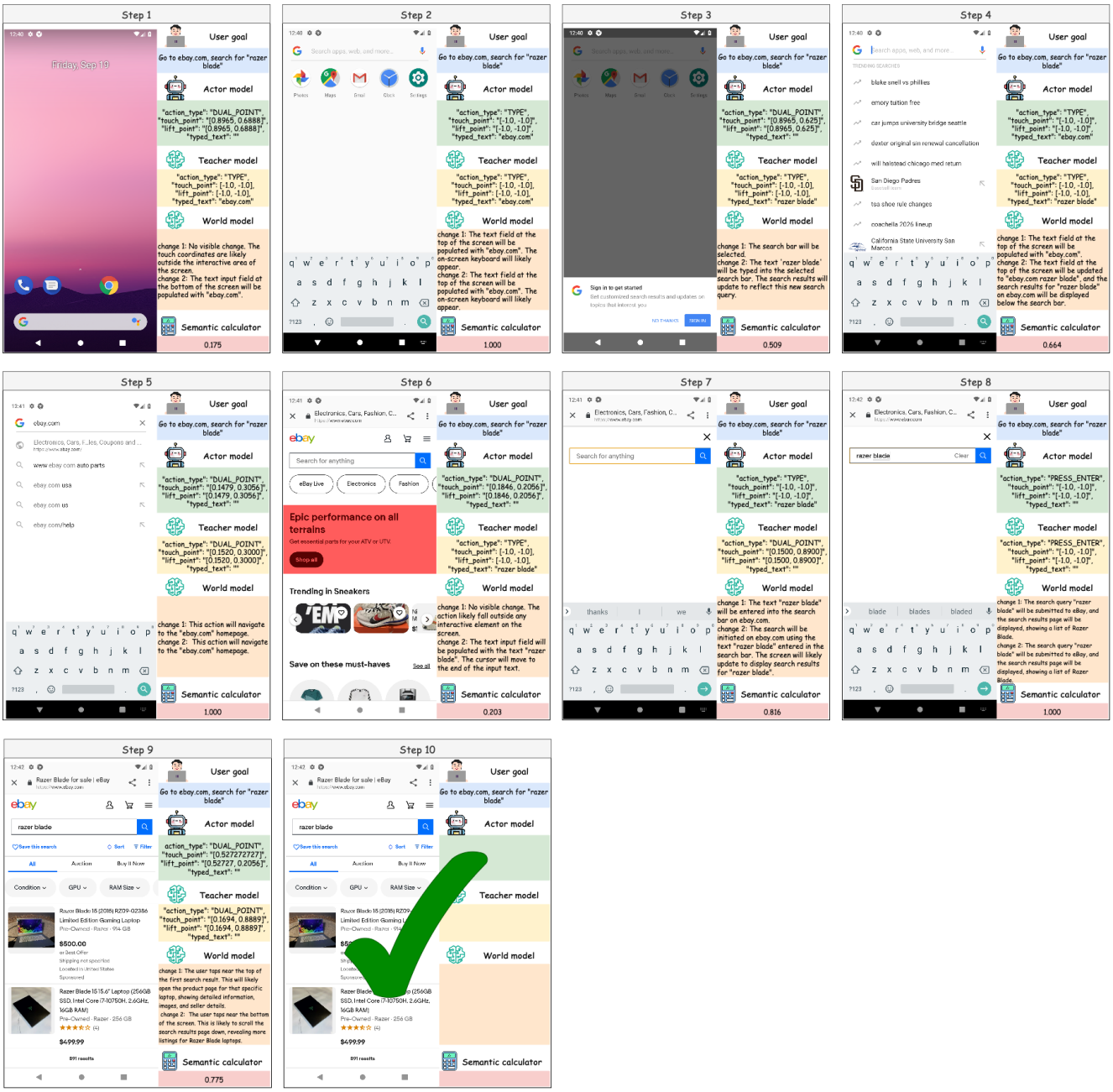


Figure 5. Example of a successful case on an AitW task with our SEE module.

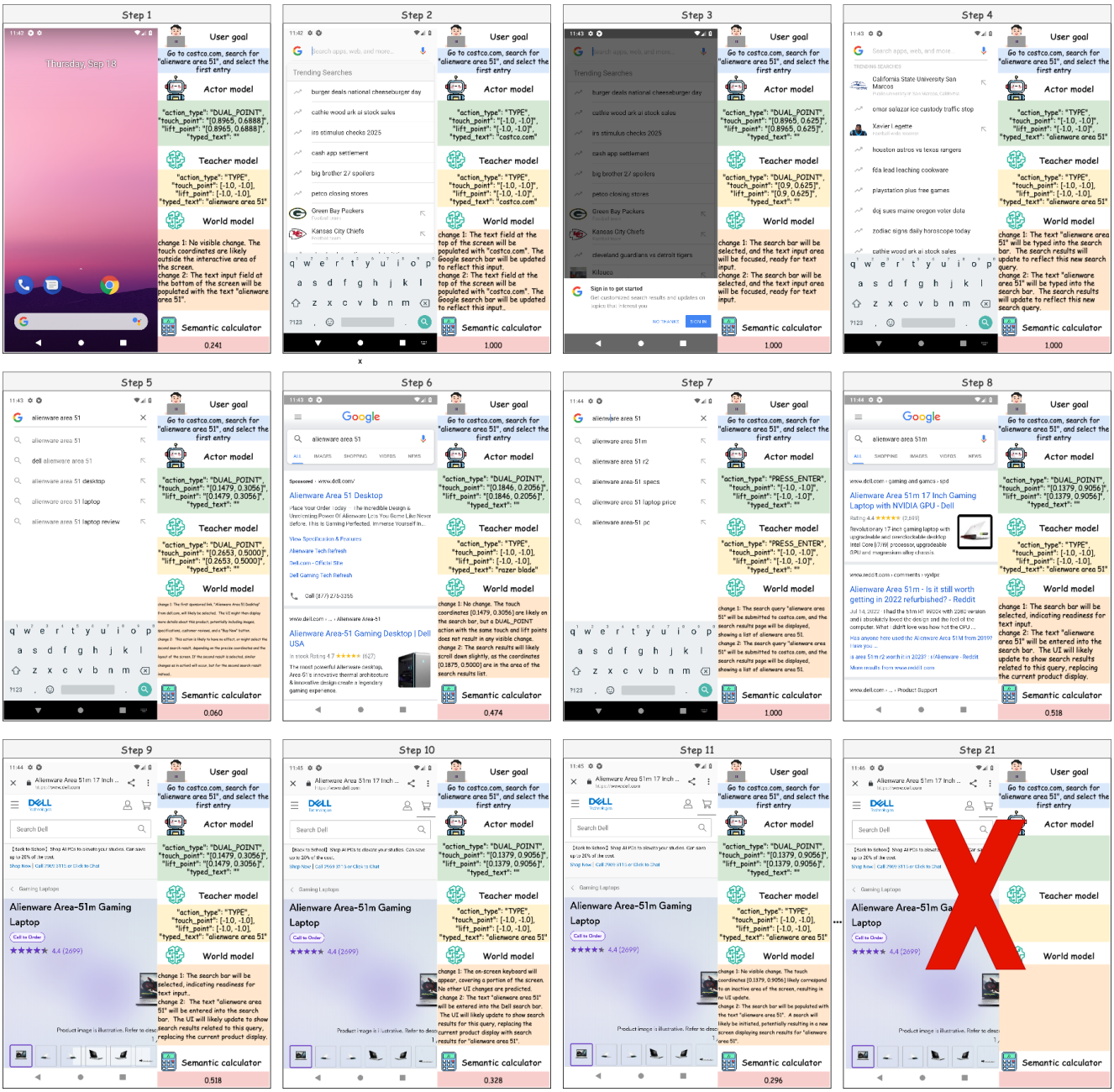


Figure 6. Example of a failure case on an AitW task with our SEE module.